UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"SOFTWARE DE SIMULACIÓN DE PULIDO DE SUPERFICIES ÓPTICAS UTILIZANDO EL MÉTODO CLÁSICO"

TESIS PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

ABRIL HERNÁNDEZ VELASCO

DIRECTORES DE TESIS

DR. JORGE GONZÁLEZ GARCÍA

M. C. C. ERIKA VERA DÍAZ

DR. AGUSTIN SANTIAGO ALVARADO

HUAJUAPAN DE LEÓN, OAX., DICIEMBRE DE 2005

Dedicatoria

A mi abuelita Hermila que donde quiera que esté sigue preocupándose y cuidando de mí y a mi madrecita linda Gisela, que sin su apoyo no hubiera salido adelante. Gracias por darme toda su confianza y amor, las quiero mucho.

Agradecimientos

Al Dr. Jorge González García y M. C. C. Erika Vera Díaz por su dirección, apoyo y consejos, mi más sincero agradecimiento.

Al Dr. Agustin Santiago Alvarado por no desampararme en mis momentos de desesperación y por los consejos que me ofreció, gracias por su valiosa amistad.

A los maestros Mario Alberto Moreno Rocha, Hilda Caballero Barbosa, Alejandro López López y Felipe de Jesús Rivera López por las aportaciones hechas a este trabajo y valiosos consejos, sinceramente gracias.

Al CONACYT por la confianza y apoyo brindados en el desarrollo de este proyecto perteneciente a “Pulido Predecible” con clave de registro 44715-F (2003).

Al Lic. Carlos E. Santibáñez Moran y especialmente a su esposa Dr. Patricia Camarillo Salvatori por brindarme apoyo en los momentos más difíciles de mi carrera, gracias de todo corazón.

A Carlos M. Santibáñez Camarillo por brindarme su cariño y estar brindarme tu apoyo en todo momento, te quiero mucho flaquito.

A dos de las personas más importantes de mi vida, mis hermanos Emilio y Yasser que han depositado toda su fe en mí y me han apoyado cuando más los he necesitado, los quiero mucho hermanitos.

Finalmente, a todas aquellas personas que influyeron directa o indirectamente en mi formación, familiares, amigos y maestros, mis más sinceros agradecimientos.

Resumen

Este documento presenta el primer software en México que simula el pulido clásico de superficies ópticas usando el Proceso Unificado de Rational (RUP), Lenguaje de Modelado Unificado (UML) y pruebas de usabilidad; para producir un software libre de alta calidad y fácil manejo. Este trabajo analiza y describe las fases de desarrollo de software con la intención de producir un software que proporciona soporte técnico dentro de los talleres de óptica Mexicanos. Además, el software permite la determinación de parámetros funcionales necesarios para producir el perfil deseado y para llevar a cabo fácilmente el proceso de generación de una superficie óptica. En trabajos futuros, este software será parte del proceso de automatización de pulido de superficies ópticas en talleres de óptica Mexicanos.

Abstract

This document presents the first software in México to simulate the classical polishing of optical surfaces, using Rational Unified Process (RUP), Unified Modeling Language (UML) and usability tests; in order to produce a free program of high quality and easy management. This work analyses and describes the phases of development of the software with the intention of producing software which gives technical support within Mexican optical shops. On the other hand, the software permits the determination of functional parameters necessary to produce the desired profile in the optical surface, and to easily carry out the process of generation. In future works, this software will be a part of the process of automatization of optical polishing surfaces in Mexican optical shops.

Índice

Lista de Figuras

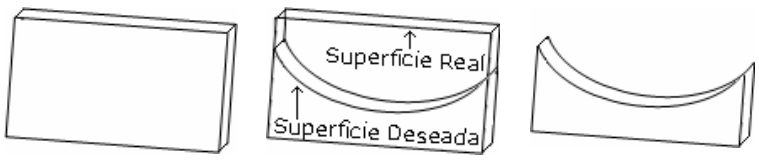
Lista de Tablas

Capítulo 1. **Introducción**

Los sistemas ópticos son utilizados en la vida cotidiana e investigación, con el objetivo de obtener una imagen. Algunos de estos instrumentos son cámaras, videocámaras, telescopios, interferómetros, espectroscopios y microscopios los cuales utilizan lentes o espejos de vidrio que deben ser generados con una calidad menor a media longitud de onda de luz para su correcto funcionamiento. Por ejemplo, es sabido que en un telescopio Newtoniano el espejo primario debe tener una forma parabólica, es decir debe tener una forma predeterminada, de lo contrario no formará las imágenes correctamente. De aquí la importancia que tiene el proceso de pulido de superficies de vidrio.

1.1 Concepto de Pulido

El pulido de superficies ópticas consiste en remover material de vidrio, de una superficie real como se observa en la Figura 1.1.a para obtener la forma de la superficie deseada como se muestra en la Figura 1.1.c.



(a) (b) (c)

Figura 1.1 Proceso de Pulido. (a) Superficie Real, (b) Material que se desea Remover, (c) Superficie Deseada.

En la actualidad, existen diferentes tipos de tecnologías utilizadas en el pulido de superficies ópticas, desde el pulido clásico hasta el de bombardeo iónico [1]. Las diferentes técnicas son utilizadas de acuerdo a los avances tecnológicos y el presupuesto de cada taller en el mundo; en México, debido a que no existen hasta el

momento avances tecnológicos en esta área, el proceso de pulido sigue siendo artesanal, por lo que diferentes talleres de pulido óptico utilizan el método de pulido clásico.

El pulido de superficies utilizando el método clásico consiste en la rotación y desplazamiento sobre el vidrio de una herramienta montada en el brazo de la máquina pulidora, manteniendo entre la herramienta y el vidrio una suspensión de abrasivo que no es más que una mezcla de pulidor u óxido de cerio con agua [1]. En la Figura 1.2, se puede observar el modelo de una máquina pulidora utilizada comúnmente en los talleres de pulido en México.

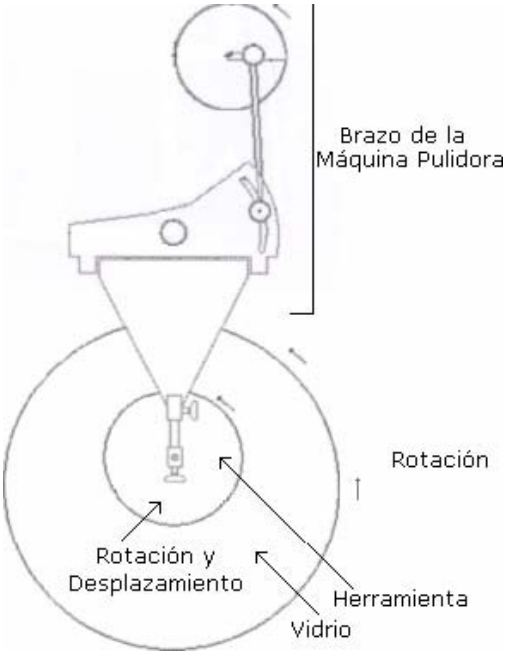


Figura 1.2 Esquema del proceso de pulido clásico.

En la Figura 1.3, se muestra una máquina de pulido clásico propiedad de la Facultad de Ciencias Físico Matemáticas de la Benemérita Universidad Autónoma de Puebla (BUAP), en donde se observa que la superficie de vidrio esta situada en un contenedor y se está llevando a cabo el proceso de pulido.

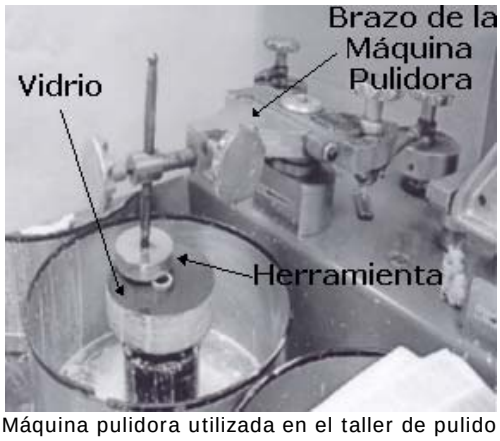
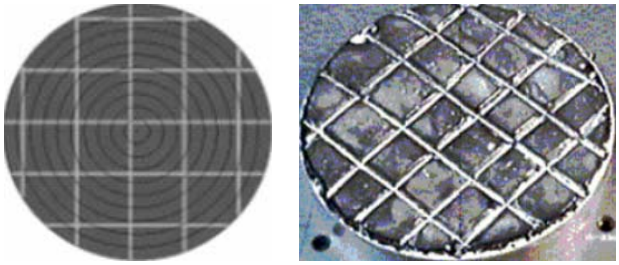


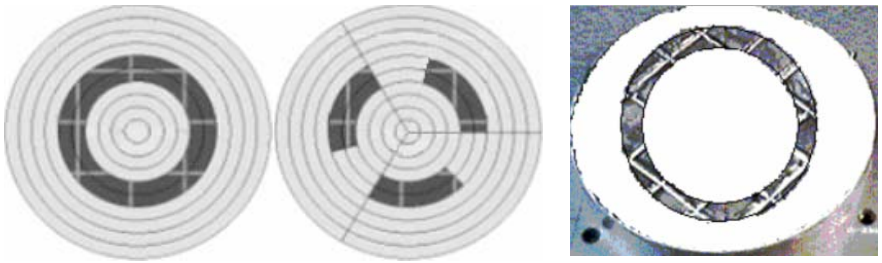
Figura 1.3 Máquina pulidora utilizada en el taller de pulido de la BUAP.

En el pulido clásico de superficies ópticas son utilizadas regularmente tres tipos de herramientas: sólida, anular y de pétalo (Figura 1.4, 1.5 y 1.6 respectivamente); estas herramientas están constituidas por anillos concéntricos, dependiendo del tipo de herramienta, estos anillos pueden ser completos o incompletos. Para cuantificar el tamaño de un anillo incompleto se mide el ángulo que forma; a este ángulo se le llama "tamaño angular del anillo incompleto". La herramienta de pétalo es llamada así, debido a que está formada por anillos incompletos que le dan la forma de pétalos, ver Figura 1.6.



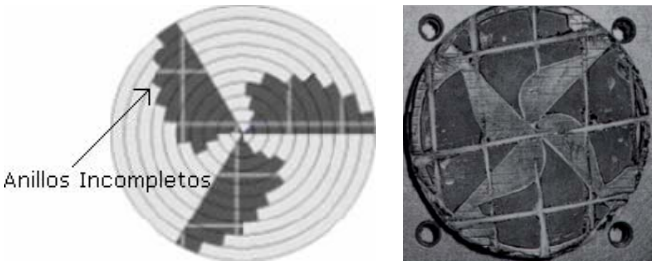
a) b)

Figura 1.4 Herramienta Sólida, a) Esquemática, b) Real.



a) b)

Figura 1.5 Herramienta Anular, a) Esquemática, b) Real.



a) b)

Figura 1.6 Herramienta de Pétalo, a) Esquemática, b) Real.

A continuación se describen los problemas presentados en el pulido clásico de superficies ópticas.

1.2 Problemática Presentada en el Pulido de Superficies Ópticas en México

En los talleres de óptica mexicanos, el proceso de pulido es totalmente artesanal ya que los técnicos hacen que las herramientas (utilizando la máquina pulidora) tallen en diferentes zonas del vidrio y por medio de su experiencia, determinan los intervalos de tiempo que la superficie permanecerá desgastándose y el momento en que este proceso debe detenerse, para generar la forma de la superficie deseada. El proceso artesanal se realiza de la siguiente manera:

- El técnico monta una herramienta sólida, generalmente del mismo tamaño del vidrio, en el brazo de la máquina pulidora, asimismo monta el vidrio en el contenedor de la misma máquina.
- Después, pone a trabajar la máquina pulidora hasta generar una esfera que se asemeje a la superficie cónica deseada, la cual es llamada esfera de referencia.
- Posteriormente, el técnico hace trabajar otra herramienta sobre cada zona de la superficie, generalmente 1/5 del tamaño del vidrio a pulir (ver Figura 1.7), el tiempo que él considere necesario.

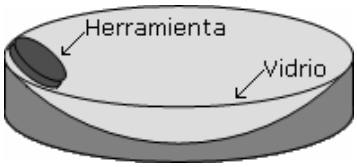


Figura 1.7. Proceso de Pulido en la Actualidad.

- El técnico desmonta la pieza de vidrio en varias ocasiones para verificar por medio de pruebas ópticas [2] si se obtuvo el perfil deseado. Si aún no lo obtiene, vuelve a pulir hasta obtener dicho perfil.

Dado este proceso, al no tener suficiente o ninguna experiencia, los problemas que surgen son los siguientes:

- Se generan pérdidas de tiempo y dinero, ya que el montar y desmontar en más de una ocasión el vidrio para verificar el desgaste obtenido, se consume tiempo innecesario al alinear la pieza de vidrio, y si se desgasta más de lo que se requiere, éste se desecha originando gastos excesivos al comprar una nueva pieza e invertir tiempo de trabajo al realizar el proceso nuevamente.
- Por otra parte, cuando dos técnicos por medio de su experiencia desean generar el mismo desgaste sobre diferentes superficies, puede ocurrir que 1) un técnico tarde más tiempo que el otro o que 2) obtengan diferentes desgastes, haciendo que varíen los resultados; por esto es necesario que los técnicos lleven un mismo procedimiento para que el tiempo y los resultados sean similares.

Los problemas mencionados hacen necesario minimizar las pérdidas de tiempo y dinero que se presentan en el pulido de superficies ópticas en México, así como también, generalizar el procedimiento que llevan a cabo los técnicos al generar una superficie.

A continuación, se describe el trabajo de investigación realizado hasta el momento.

1.3 Trabajo Previo

Los doctores Alberto Cordero Dávila y Carlos Robledo, profesores investigadores de la BUAP, crearon en Fortran para Linux dos programas independientes que simulan desgastes *arbitrarios* (dependientes de los parámetros de pulido) utilizando herramientas sólidas, anulares y de pétalos, empleando la ecuación de Preston [3], los cuales son la base de este proyecto de tesis.

Con el programa de la herramienta sólida utilizando una herramienta pequeña sobre cada zona de la superficie a pulir se puede generar el desgaste *deseado,* dejando en cada zona la herramienta un determinado tiempo. Este tiempo depende de la experiencia del usuario lo que nos lleva a que el método empleado es experimental.

En el programa de la herramienta de pétalo por medio de prueba y error el usuario puede diseñar la forma de la herramienta para generar un desgaste *deseado*. Para poder hacer esto, se requiere de cierta experiencia del usuario.

Es importante señalar que este proyecto de tesis es parte del proyecto existente entre la Universidad Tecnológica de la Mixteca (UTM) y la BUAP llamado “Pulido Predecible”, financiado por CONACYT con clave de registro 44715-F (2003), en el cual se desarrollarán los puntos que se presentan en la sección 1.5.

1.3.1 Ecuación de Preston Aplicada al Pulido de Superficies

En 1927 Preston publicó por primera vez la “*Teoría y Diseño de las Máquinas de Pulido para Placas de Vidrio*” [4]. A continuación, se detalla la forma en la que la ecuación de Preston fue aplicada para obtener el desgaste simulado. En la Figura 1.8, se muestra una herramienta de pulido ejerciendo presión sobre una superficie de vidrio.

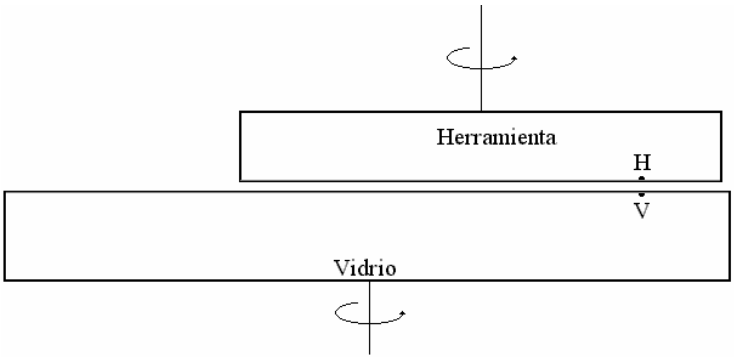


Figura 1.8 Puntos en contacto entre el vidrio *V* y la herramienta *H*.

De acuerdo a la figura anterior, denominamos H al punto de contacto sobre la herramienta, V al punto correspondiente sobre el vidrio y denotamos por p y v a la presión y la velocidad relativa entre los puntos H y V respectivamente. Preston afirmó que existe evidencia experimental para creer que la cantidad de pulido h, producido durante el intervalo de tiempo (o, τ) es proporcional a la ecuación 1.1:

$$h = \int_0^{\tau} ACpv \cdot dt \qquad (1.1)$$

donde A es una constante que depende de factores tecnológicos como el tipo de vidrio, composición del abrasivo, material de la herramienta, temperatura, humedad, entre otros y C es una función que define la existencia ($C=1$) o no ($C=0$) de contacto entre el vidrio y la herramienta para cada instante de tiempo t.

La ecuación anterior no es tan fácil de aplicar, ya que:

1. Un punto del vidrio no siempre está en contacto con la herramienta;
2. La velocidad relativa, para un mismo instante de tiempo, es diferente para los diferentes puntos del vidrio, o la presión varía de un punto a otro de la superficie cuando la herramienta sale parcialmente del vidrio.

Por lo tanto, la evaluación de la integral de la ecuación (1.1) debe realizarse de tal manera que pueda ser calculado por una computadora. A continuación, se describe un procedimiento para calcularla.

El intervalo de tiempo (0, τ) es dividido en N subintervalos de tiempo con duración:

$$\Delta t = \frac{\tau}{N} \tag{1.2}$$

El instante de tiempo en el que inicia cada subintervalo esta dado por

$$t_i = i \cdot \Delta t \tag{1.3}$$

donde *i = 0, 1, 2, 3, ..., N-1*

En cada instante t_i se evalúan C, p y v asumiendo que se mantienen constantes durante todo el subintervalo. La integral es calculada mediante la aproximación de la ecuación 1.4.

$$\int_0^{\tau} ACpv \cdot dt = A\Delta t \sum_{i=0}^{N-1} C_i p_i v_i \tag{1.4}$$

donde el subíndice indica el valor de la variable respectiva en el intervalo i-ésimo.

1.4 Trabajo Relacionado

Catedráticos de la BUAP han desarrollado investigaciones con la herramienta de pétalo y han constatado que al variar la forma de sus anillos se obtienen diferentes desgastes. En la actualidad, se trabaja de manera experimental con una herramienta de tamaño similar a la superficie (ver Figura 1.9), que al pulir en determinado tiempo, genera el desgaste deseado.

Figura 1.9. Pulido Clásico con una Herramienta de Pétalo.

Pero también, es necesario ajustar los tamaños angulares de los anillos para que formen la herramienta de pétalo y al trabajar con ésta obtener el desgaste que se requiere.

1.5 Propuesta de Tesis

Cabe resaltar que no todos los talleres de pulido en México cuentan con sistemas de cómputo que trabajen en la plataforma Linux en la que fue implementada la ecuación de Preston, por lo que es necesario realizar un software de simulación multiplataforma, es decir, que se ejecute en Windows y Linux.

Debido a la problemática en el proceso de pulido y partiendo de los trabajos relacionados se pretende crear un programa amigable y usable, que integre los programas del trabajo previo en plataforma Linux y trasladarlos a C++ Builder 6.0 (Windows) y Kylix 3.0 Enterprise (Linux). Que además, permita generar desgastes deseados que no dependan de la experiencia del usuario. Por lo que se propone aplicar:

1. Mínimos cuadrados para calcular los tiempos de estancia de una herramienta sólida pequeña para generar el desgaste deseado.
2. Algoritmos genéticos para calcular la forma de las herramientas de pétalo que generen el desgaste deseado.

1.6 Alcances del proyecto de tesis

Este software servirá de consulta y apoyo a los talleres de la BUAP y el INAOE, proporcionando a los técnicos una forma *cualitativa* del desgaste (perfiles obtenidos sin unidades de medición) que pueden obtener al utilizar las tres herramientas de pulido antes mencionadas.

Para interpolar las simulaciones (obtenidas con este software) con las pruebas experimentales, es necesario obtener un factor de calibración el cual se investigará en trabajos futuros. Como consecuencia, este software no será del tipo empotrado.

1.7 Objetivo General

Analizar, diseñar e implementar un software de simulación de pulido clásico (SSPC) multiplataforma.

1.7.1 Objetivos Específicos

- Aplicar un proceso y metodología de desarrollo de software para garantizar la *calidad y usabilidad* del SSPC.
- Implementar el SSPC en plataforma Windows y Linux.
- Calcular los tiempos de estancia que una herramienta sólida debe estar sobre cada zona de la superficie de vidrio y así generar el desgaste deseado utilizando mínimos cuadrados.
- Diseñar la forma de la herramienta de pétalo utilizando como método de optimización algoritmos genéticos, para obtener el desgaste deseado sobre una superficie.

1.7.2 Objetivos Didácticos

- Aprender y aplicar (a un problema real) las técnicas de optimización de algoritmos genéticos y mínimos cuadrados.
- Aprender y aplicar conceptos del área de pulido de superficies.

Finalmente, se describe a continuación cada uno de los capítulos que constituyen este proyecto de tesis.

1.8 Esquema de la Tesis

El capítulo 2 se dedica al estudio de las metodologías utilizadas para el desarrollo de software: detección de objetos y clases, y análisis de requerimientos.

El capítulo 3 describe la implementación de mínimos cuadrados para obtener los tiempos de estancia de una herramienta sólida, sobre la superficie de vidrio, para obtener el desgaste deseado.

El capítulo 4 detalla la implementación de algoritmos genéticos para encontrar el perfil óptimo de la herramienta de pétalo, que genere el desgaste deseado.

En el capítulo 5 se analizan las pruebas de usabilidad realizadas al software de simulación.

El capítulo 6 muestra los resultados obtenidos con el simulador de pulido propuesto en esta tesis.

El capítulo 7 presenta las conclusiones y trabajo futuro.

El apéndice A describe el proceso llevado a cabo para migrar el simulador de Windows a Linux.

Apéndice B contiene el diagrama de casos de uso unificado.

El apéndice C describe cada uno de los casos de uso.

El apéndice D presenta el diagrama de clases unificado del SSPC.

El apéndice E describe cada una de las clases utilizada en el SSPC.

El apéndice F contiene las plantillas de los cuestionarios utilizados para la realización de las pruebas de usabilidad.

Finalmente, el apéndice G contiene los reconocimientos obtenidos por trabajos relacionados a este proyecto de tesis.

Capítulo 2. **Análisis y Diseño del Simulador de Pulido**

2.1 Introducción

En el presente capítulo se describe el proceso de ingeniería de software aplicada al SSPC propuesto en este proyecto de tesis, con el que se pretende apoyar a técnicos mexicanos, poniendo a su disposición una interfaz amigable y usable que les permita obtener tiempos de estancia y desgastes deseados.

La ingeniería de software está relacionada con las teorías, métodos y herramientas para el desarrollo o mejoramiento profesional de software. El proceso de ingeniería de software se define como un conjunto de etapas parcialmente ordenadas con la intención de lograr un “software de calidad”. Es importante remarcar que el software además de calidad, debe ofrecer al usuario la funcionalidad y el rendimiento requerido, esto es, que sea mantenible, es decir, el software debe evolucionar para adaptarse a las necesidades cambiantes; fiable, realizar su objetivo; eficiente, no debe malgastar o hacer mal uso de los recursos del sistema y usable, es decir, de fácil uso para los usuarios [5,6].

Los principios de la ingeniería de software alcanzan tanto al proceso, como al producto final, ya que describen las cualidades deseables en abstracto y el plan del proyecto. En la siguiente sección se presenta el proceso seleccionado.

2.1.1 Selección de un Proceso del Software

El proceso de desarrollo de software establece los pasos que se deben seguir con procedimientos y técnicas para la producción del mismo. Consta de una secuencia de pasos combinando procedimientos de gestión, métodos técnicos y soporte automatizado para generar productos de software. Además, para aplicar los principios

de la ingeniería de software, el desarrollador debe emplear la metodología apropiada y las técnicas específicas que le ayuden a incorporar las propiedades deseables en los procesos y productos [7].

Un modelo de proceso de software es una representación simplificada del mismo, pero es presentado desde una perspectiva concreta; ejemplos de estas perspectivas son perspectiva de flujo de trabajo (*workflow)* o secuencia de actividades, perspectiva de flujo de datos (*data flow*) y perspectiva de rol/acción. Entre los modelos de proceso genéricos se encuentra el modelo de cascada, de espiral, iterativo, construcción de prototipos, integración a partir de componentes reutilizables, entre otros. De estos modelos es posible elegir una combinación de fases o ciclo de vida completo [5].

Uno de estos modelos mencionados es el Proceso Unificado, el cual es iterativo e incremental, y que se adapta a través de los proyectos con respecto a su tamaño y complejidad. El proceso unificado de la compañía de Rational (RUP) se basa en muchos años de experiencia en el uso de la tecnología orientada a objetos para el desarrollo de software, donde confluyen tres autores: Grady Booch, James Rumbaugh e Ivar Jacobson. El Proceso Unificado de Rational (RUP) ha adoptado un enfoque que se caracteriza por los siguientes puntos:

1. Interacción continua con el usuario desde un inicio.
2. Mitigación de riesgos antes de que ocurran.
3. Aseguramiento de la calidad.
4. Involucramiento del equipo en todas las decisiones del proyecto.
5. Anticiparse al cambio de requerimientos.
6. Centrarse en la arquitectura.
7. Guiado por casos de uso.
8. Confrontación de riesgos

De los cuales es importante resaltar los puntos 3, 6 y 7 que son de interés para los objetivos de esta tesis.

Por otra parte, Rational Unified Process (RUP) esta formado por fases o ciclos de vida del software que son denominadas concepción, elaboración, construcción y transición. La concepción es especificar el alcance del proyecto y definir el caso de uso.

La elaboración es proyectar un plan, establecer las características y cimentar la arquitectura. La construcción es crear el producto y la transición es transferir el producto a sus usuarios [8].

Cada una de estas fases está dividida en 9 flujos de trabajo, 6 flujos de trabajos principales (ingeniería) y 3 secundarios (soporte). Los flujos principales son modelado del negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue. Los flujos secundarios son seguimiento del proyecto, control de cambios y configuración y ambiente. Para el desarrollo del SSPC se pretende llevar acabo estas fases y los flujos de trabajo.

A continuación, se define el lenguaje de modelado del software que fue utilizado en este proyecto.

2.1.2 Selección de Lenguaje de Modelado del Software

El desarrollo de los lenguajes de ingeniería de software fue propiciado por la necesidad de atender problemas específicos de la producción de software cada vez más complejos. A lo largo del tiempo, dichos lenguajes han evolucionado hasta consolidar el Lenguaje de Modelado Unificado (ver Tabla 2.1).

Unified Modeling Language (UML) es una notación que evolucionó a partir del diseño basado en Booch, Rumbaugh y Jacobson. Es el lenguaje orientado a objetos más reciente para el desarrollo de software, aceptado por el Unified Modeling Languaqe Specification (OMG), además de ser el lenguaje estándar para RUP para la visualización, especificación, construcción y documentación de artefactos del software [9], por lo cual, fue elegido para el modelado del SSPC. Cabe señalar, que dadas las características de UML se adoptó una metodología orientada a objetos para la abstracción de clases de este simulador.

Tabla 2.1 Cronología de Métodos [10].

Año		Metodología	Autor
1991	OMT	Técnica de Modelado de Objetos *(Object Modeling Technique)*	Rumbaugh *et al.*
1992	OL	Ciclos de Vida de los Objetos	Shaler & Mellor
1992	OOSE	Ingeniería de Software Orientada a Objetos *(Object Oriented Software Engineering)*	Jacobson *et al.*
1994	OOADA	Análisis y Diseño con Aplicaciones Orientadas a Objetos *(Object-Oriented Analisys and Design with Applications)*	Booch
1994	MOSES	Metodología para la Ingeniería Software de Sistemas Orientadas a Objetos *(Methodology for Object-Oriented Software Engineering of Systems)*	Henderson-Sellers y Edwars
1994	Fusión	Método Fusión	Coleman *et al.*
1996	UML	Lenguaje de Modelado Unificado *(Unified Modeling Language)*	Booch-Rumbaugh-Jacobson (Rational Software)

En el siguiente apartado se describen brevemente algunas tecnologías de programación y la seleccionada para el desarrollo de este proyecto.

2.1.3 Selección de la Técnica de Programación

Actualmente, existen diferentes tecnologías de programación tales como la programación **procedimental**[1], **lógica**, **orientada a objetos**, entre otras [11]. La programación orientada a objetos tiene como características esenciales la abstracción, **encapsulamiento**, **herencia** y **polimorfismo** que pueden ser modelados con UML. Además, la programación orientada a objetos permite las implementaciones puedan ser mantenibles, adaptables y rediseñables continuamente (reutilización del código), características que otras tecnologías no soportan [12]. Por consiguiente, la técnica de programación aplicada al SSPC es orientada a objetos.

[1] Todas las palabras en negritas que son utilizadas en este documento están definidas en el Glosario.

A continuación, se describe el proceso de desarrollo que se llevó a cabo en el simulador de pulido clásico de superficies ópticas.

2.2 Proceso de Desarrollo

El modelo de negocio diseñado para desarrollar el SSPC se describe en la Figura 2.1. Este modelo muestra las actividades a realizar en esta tesis.

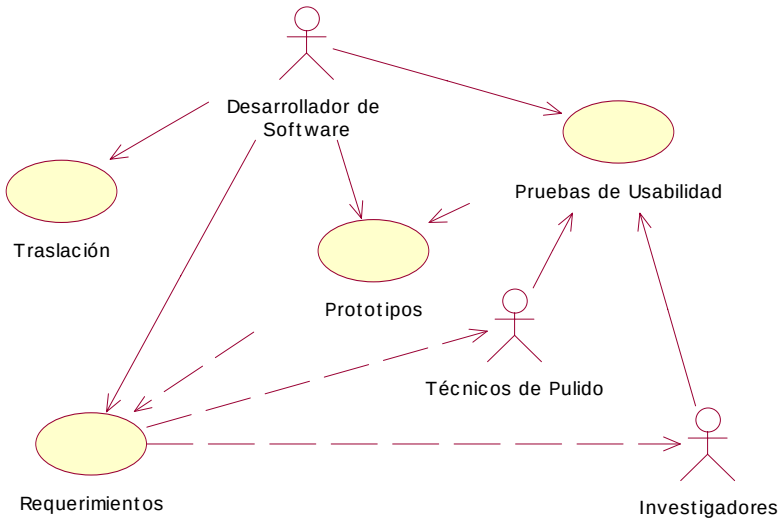


Figura 2.1 Modelo de negocios de casos de uso.

El diagrama que muestra el comportamiento del proceso de desarrollo del software se observa en la Figura 2.2. Este es un diagrama de estados en la que se simplifican las fases de RUP.

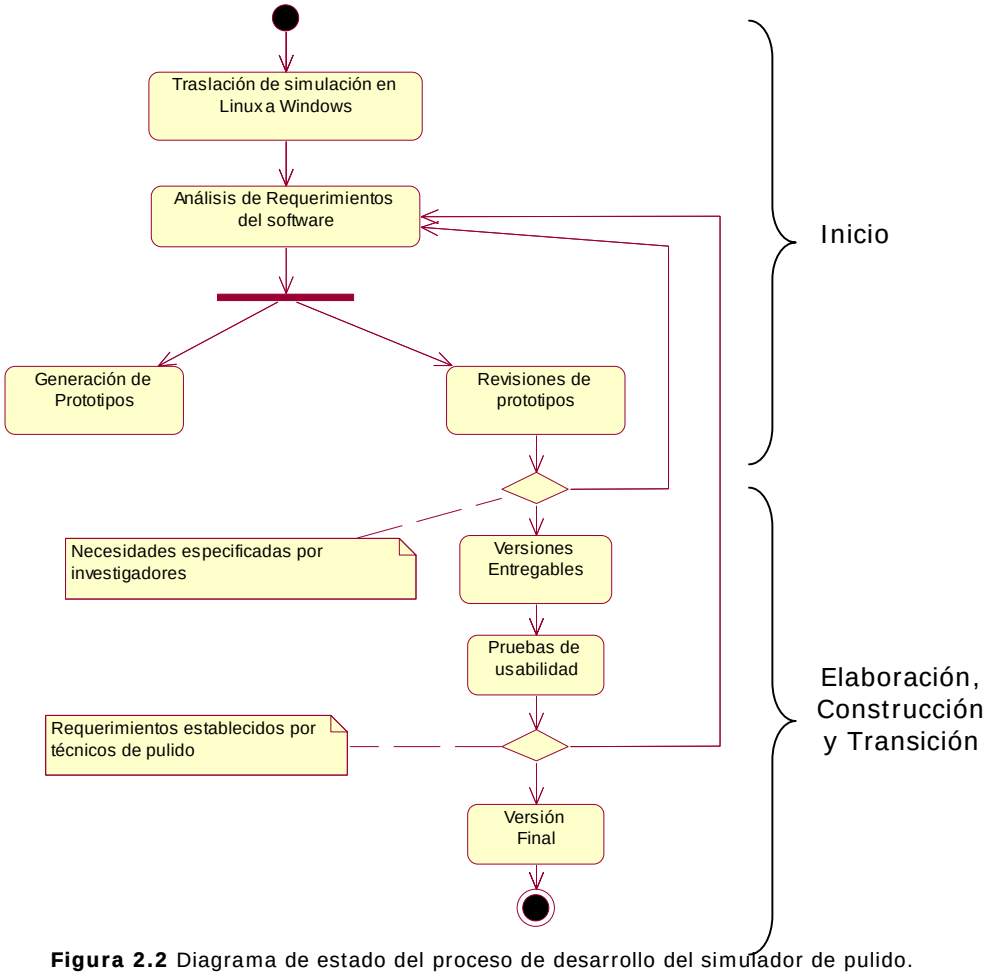


Figura 2.2 Diagrama de estado del proceso de desarrollo del simulador de pulido.

Como se mencionó en la sección 1.5, ya existía una aplicación para Linux, escrito en Fortran, que simulaba el proceso de pulido, la cual fue trasladada línea a línea al lenguaje de programación C++ haciendo uso del visual C++Builder 6.0. Al finalizar la migración, se inició el análisis y documentación de requerimientos. Los requerimientos establecidos por los investigadores se presentan en la sección 2.2.1. En la sección 2.2.2 se muestra la evolución y revisión de los prototipos del simulador así como también se mencionan las pruebas de usabilidad realizadas a este software.

La comunicación con los usuarios se llevó a cabo como lo muestra la Figura 2.3.

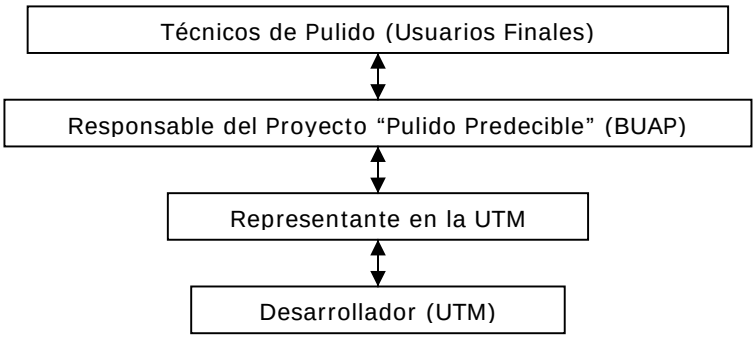


Figura 2.3 Comunicación Usuario – Programador.

2.2.1 Requerimientos

Inicialmente, los **requerimientos** fueron obtenidos directamente de los profesores representantes en la UTM quienes son el Dr. Jorge González García y el Dr. Agustin Santiago Alvarado del Instituto de Física y Matemáticas. Posteriormente, de los técnicos de la BUAP y del Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE) de acuerdo a las pruebas de usabilidad aplicadas al SSPC (ver capítulo 5). Se realizaron cuatro iteraciones para la obtención de requerimientos a lo largo del proceso, obteniendo finalmente la lista que se presenta a continuación.

1. Simulación del Proceso de Pulido

1.1 El usuario podrá elegir con qué tipo de herramienta desea trabajar y de acuerdo a la herramienta elegida ya sea sólida, anular o de pétalo.

1.2 El software solicitará los parámetros de pulido correspondientes:

- 1.2.1 De acuerdo al tipo de herramienta seleccionada: Radio interno, radio externo, velocidad angular de giro y velocidad angular de oscilación.
- 1.2.2 Datos del vidrio: velocidad angular de giro y radio.

1.3 El software ofrecerá como salida:

- 1.3.1 Una gráfica de desgaste.
- 1.3.2 Gráfica de la herramienta.
- 1.3.3 Oscilación del borde de la herramienta sobre el vidrio.
- 1.3.4 Un archivo de datos que contiene el desgaste obtenido de la simulación.

2. Cálculo de Tiempos de Estancia

2.1 El software requerirá los parámetros correspondientes a:

- 2.1.1 Datos del vidrio: constante de conicidad y radio de curvatura.

2.2 El software proporcionará como salida:

2.2.1 Gráfica de tiempo contra posición sobre el vidrio.

2.2.2 Gráfica de desgaste, perfil obtenido.

2.2.3 Un archivo de datos que contiene los tiempos obtenidos en la simulación.

3. Diseño de la herramienta de pétalo para obtener el desgaste deseado

3.1 El software solicitará al usuario los parámetros correspondientes a:

3.1.1 Datos de la herramienta de pétalo: Radio, velocidad angular de giro y velocidad angular de oscilación.

3.1.2 Datos del vidrio: velocidad angular de giro y radio.

3.1.3 Datos del desgaste deseado: simulado o por archivo de datos, constante de conicidad y radio de curvatura.

3.2 El software ofrecerá como salida:

3.2.1 Una gráfica de desgaste generado contra el desgaste deseado.

3.2.2 Gráfica de la herramienta de pétalo.

3.2.3 Oscilación del borde de la herramienta sobre el vidrio.

3.2.4 Un archivo de datos que contiene el desgaste obtenido de la simulación.

4. Manejo de Plataformas

4.1 El software deberá ser implementado para dos plataformas:

4.1.1 Windows (98/ME/2000/XP).

4.1.2 Linux (Suse 7.3).

5. Tutorial de Pulido.

5.1 Debido a que no todos los usuarios que utilizarán el software son experimentados en el tema de pulido de superficies, el software contará con un archivo de ayuda que por medio de ejemplos le indicará al usuario como puede llevar a cabo la simulación del proceso de pulido.

Al finalizar con la etapa de requerimientos se obtuvieron prototipos del software que a lo largo de las diversas iteraciones se convirtieron en una primera versión del simulador.

2.2.2 Prototipos y Revisión

Al trasladar la aplicación de Linux a Windows se obtuvo el primer prototipo del SSPC, el cual simulaba el proceso de pulido con tres herramientas. En la Figura 2.4 se puede observar que se tenían seis opciones 1) modificar datos de la máquina pulidora, 2) modificar datos de la herramienta sólida, 3) datos de la herramienta anular, 4) modificación de parámetros de la herramienta de pétalo, 5) opción para modificar los parámetros del vidrio y 5) iniciar la simulación del pulido clásico.

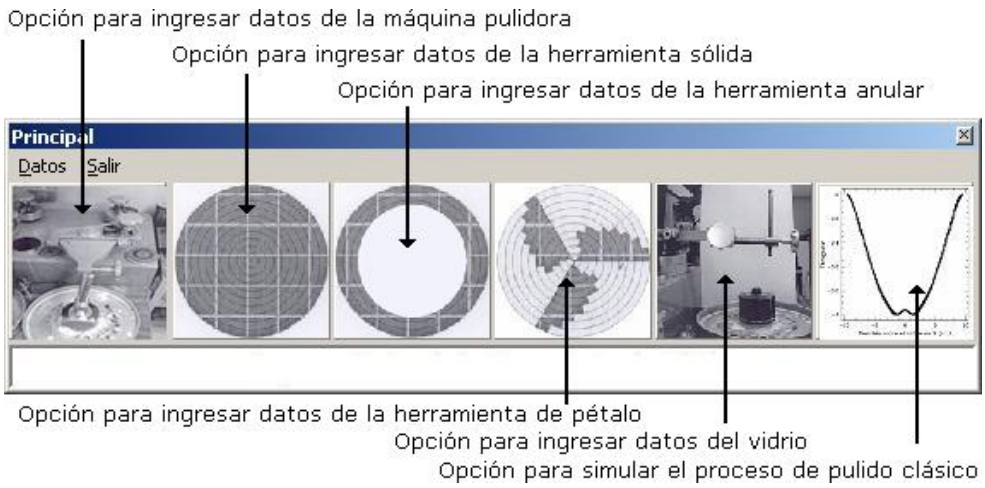


Figura 2.4 Primer prototipo del SSPC.

Las posteriores revisiones aplicadas a esta versión generaron detalles, la presentación no era entendible por el usuario y era poco usable, que fueron corregidos hasta obtener dos prototipos adicionales (Figura 2.4 y 2.5). En el segundo prototipo se simulaba con las tres herramientas sólo que los resultados se visualizaban de distinta manera, se tenía un menú en el cual el usuario podía elegir la herramienta con la que pretende trabajar, modificaba los parámetros e iniciaba la simulación, también podía visualizar el valor de los parámetros que intervienen en el proceso de simulación en la parte inferior de la pantalla (Figura 2.5).

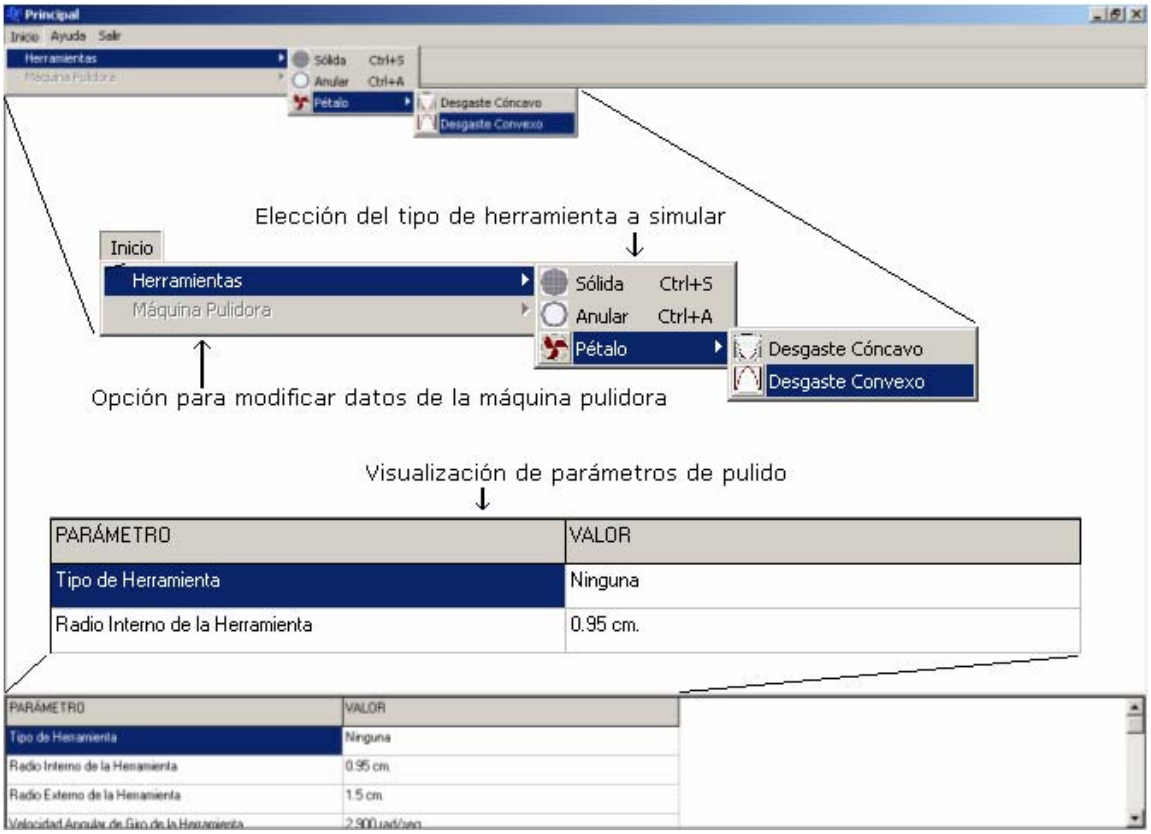


Figura 2.5 Segundo prototipo del SSPC.

El último prototipo obtenido, cumplía con todos los objetivos de este trabajo de tesis y se realizaron algunos cambios en la presentación al usuario, tal y como se muestra en la Figura 2.6.

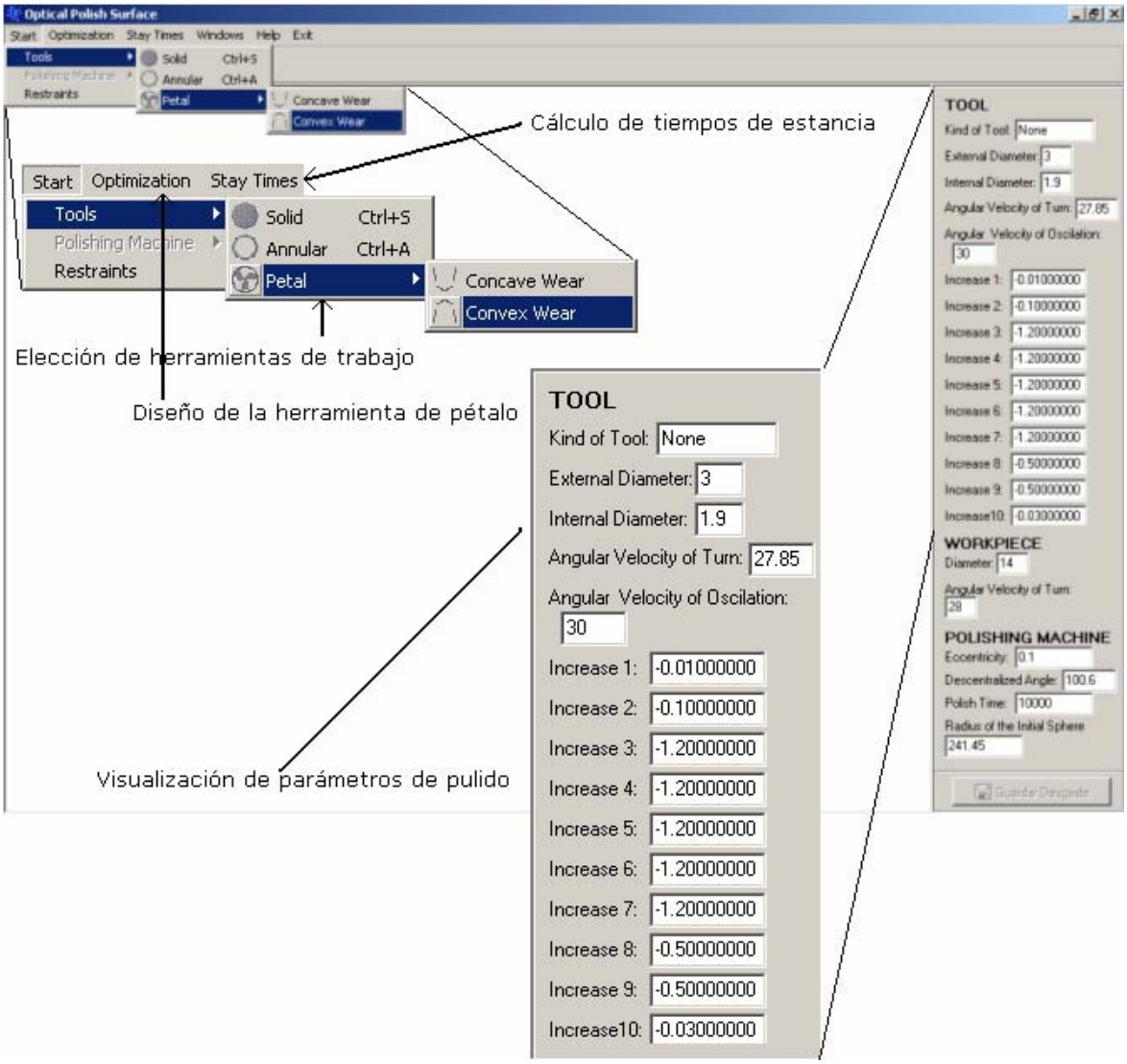


Figura 2.6 Primer versión del SSPC.

Al realizar las revisiones de los prototipos se pudo constatar que debían existir algunas restricciones en cuanto a los parámetros de pulido se refieren ya que al realizar los cálculos podían causar un error en el funcionamiento del SSPC, por lo cual dichas restricciones y nuevos requerimientos se presentan en la Tabla 2.2.

Tabla 2.2 Restricciones del simulador.

Variable	Restricción
Diámetro Externo de la Herramienta	Debe ser menor o igual al diámetro del vidrio.
Diámetro Interno de la Herramienta	Debe ser menor al diámetro externo.
Diámetro del Vidrio	Debe ser mayor que 0 y menor o igual a 30 cm.
Velocidad de Oscilación de la Herramienta	Debe ser mayor a 30 rpm y menor a 120 rpm.
Velocidad de Giro de la Herramienta	Debe ser mayor a 20 rpm y menor a 120 rpm.
Velocidad de Giro del Vidrio	Debe ser mayor a 0 rpm y menor a 120 rpm.
Amplitud de Oscilación de la Herramienta sobre el Vidrio	Debe ser menor o igual a 6 cm.

Al concluir con las revisiones y obtener una primera versión del software de simulación, se aplicaron pruebas de usabilidad (ver capítulo 5) en el Laboratorio de Usabilidad (UsaLab)[2] con técnicos e investigadores de pulido pertenecientes a la BUAP, de acuerdo a estos resultados se obtuvo una segunda versión del software, a la que también se aplicaron pruebas de usabilidad en el INAOE con técnicos de pulido que laboran en dicha institución. Finalmente, con estas pruebas se obtuvo la versión final del SSPC.

Por otra parte, como se mencionó en la sección 2.1.1, es necesario integrar elementos formales, de diferentes metodologías, presentados en el siguiente apartado.

2.3 Elementos Formales Incorporados

Algunos elementos formales incorporados son:

- Documentación UML (casos de uso, clases, diagramas de secuencia y de colaboración).
- Pruebas de usabilidad.

[2] UsaLab ubicado en la UTM, en la ciudad de Huajuapan de León, Oaxaca.

- Algoritmos genéticos.
- Mínimos cuadrados.

En la Figura 2.7, se presenta la arquitectura del SSPC, en la cual el software solicita los parámetros de pulido por medio de una interfaz (comunicación entre el usuario y el SSPC), con lo cual se realiza la simulación del proceso de pulido obteniendo como resultado gráficas de desgaste y tiempos, dicha simulación esta basada en dos plataformas Windows y Linux.

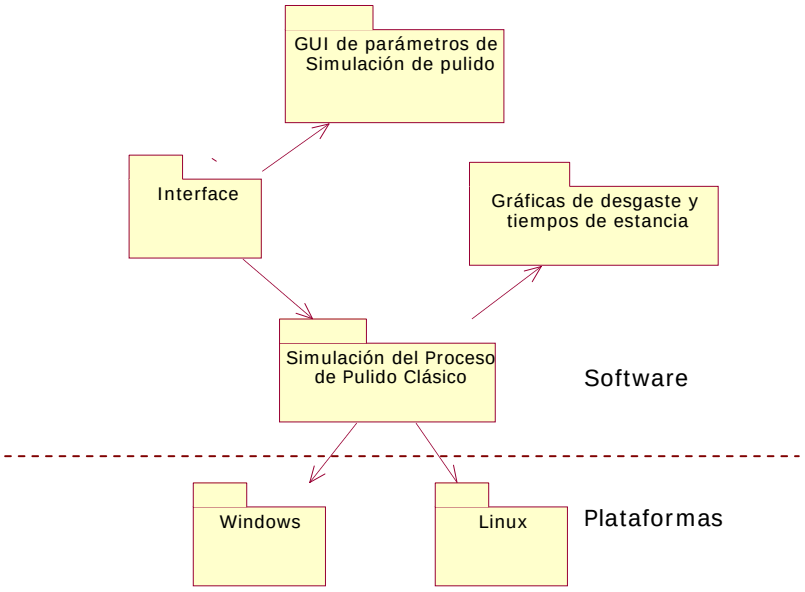


Figura 2.7 Arquitectura del SSPC.

La obtención de requerimientos fue evolucionando al igual que los prototipos. El diagrama general de casos de uso del SSPC (ver Apéndices B y C) se muestra en la Figura 2.8.

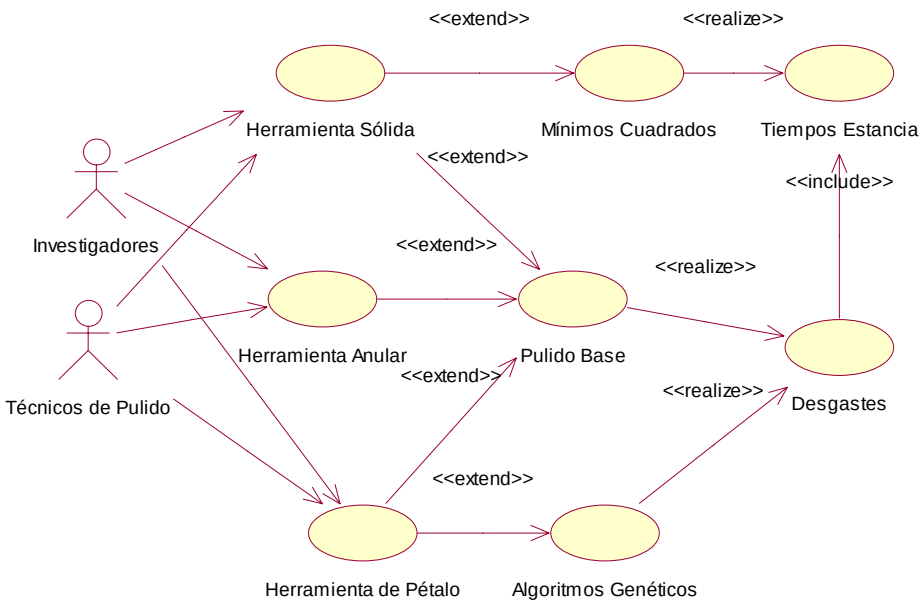


Figura 2.8 Diagrama de Casos de Uso.

Posteriormente, se identificaron cada una de las **clases** que intervienen en el funcionamiento de los casos de uso mostrados anteriormente; el Apéndice D presenta el diagrama de clases utilizado en el SSPC, en el Apéndices E, se detallan cada una de las clases.

Capítulo 3. **Aplicación de Mínimos Cuadrados al Pulido de Superficies**

3.1 Introducción

Cuando un técnico de pulido desea generar una lente con superficies cónicas, lo primero que hace es generar una superficie esférica de referencia, ésta debe ser la que mejor se ajuste a la superficie que se desea obtener, después de esto, el técnico utiliza una herramienta sólida de 1/5 del tamaño de la superficie de vidrio a fabricar (ver sección 1.2) y desgasta las zonas sobrantes, tallando con la herramienta sobre cada zona de la superficie de vidrio por determinado tiempo, el cual, suele variar dependiendo del intervalo en el cual se esté trabajando, para así obtener la superficie deseada. Por lo tanto, se pretende calcular los tiempos que la herramienta sólida debe permanecer en las diferentes zonas de la superficie a construir, para obtener el perfil deseable [13, 14], para lo cual, se utilizará la técnica de mínimos cuadrados. En este capítulo se explicará primeramente en que consiste la técnica de mínimos cuadrados y posteriormente se aplicará al problema de cálculo de los tiempos de estancia de una herramienta sólida.

3.2 Método de Mínimos Cuadrados

La técnica de mínimos cuadrados consiste en encontrar las menores diferencias al cuadrado que existan entre dos funciones ó de una distribución de puntos a una función determinada.

El método de mínimos cuadrados para resolver un problema, requiere determinar la mejor aproximación que se pueda ajustar y para ello, suma todas esas desviaciones y las guarda en una función objetivo buscando minimizarlas. Para el caso de un ajuste lineal a una distribución de puntos, el error es la suma de los cuadrados de las diferencias entre los valores de y_i' en la línea de aproximación y los valores de

y_i dados. Por tanto, hay que encontrar las constantes a y b que reduzcan al mínimo el error de mínimos cuadrados:

$$\sum_{i=1}^{10}[y_i-(ax_i+b)]^2 \tag{3.1}$$

donde $y_i'=ax_i+b$.

El método de mínimos cuadrados es el procedimiento más adecuado para determinar las mejores aproximaciones lineales [15]. Este método, concede mayor valor relativo al punto que está alejado del resto de los datos, pero no permitirá que este punto domine enteramente la aproximación.

Si en un problema la calidad del ajuste puede ser especificada por la función objetivo, dependiendo de los valores numéricos que tome y con ello nos diga que tan buena es una solución; mínimos cuadrados consistirá en minimizar la suma de los cuadrados de las f_i, es decir

$$\phi=\sum_{i=1}^{M} f_i^{\,2} \tag{3.2}$$

la ecuación (3.2) es llamada función objetivo y es la que se desea minimizar en lugar de igualarla con cero. Donde f_i depende de los parámetros del sistema a optimizar y son mediciones de las desviaciones del sistema a un valor blanco dado por

$$f_i=\left[\sum_{j=1}^{N}\left(x_j-x_{0j}\right)\right] \tag{3.3}$$

con x_j valores experimentales ó calculados y x_{0j} valores deseados o blancos.

La función de mérito ϕ definida en la ecuación (3.2) puede escribirse en notación matricial como

$$\phi=F^TF \tag{3.4}$$

donde F es un vector columna cuyas componentes son las f_i y F^T es su transpuesto.

Si hacemos un desarrollo de cada función f_i en series de Taylor, y cortamos la serie después de los términos con primeras derivadas, entonces

$$\phi = \sum_{i=1}^{M}\left[\mathrm{f}_{0i} + \sum_{j=1}^{N} \frac{\partial \mathrm{f}_{\mathrm{i}}}{\partial X_j}\left(x_j - x_{0j}\right)\right]^2 \tag{3.5}$$

con

$$f_i = \left[\mathrm{f}_{0i} + \sum_{j=1}^{N} \frac{\partial \mathrm{f}_{\mathrm{i}}}{\partial X_j}\left(x_j - x_{0j}\right)\right] \tag{3.6}$$

donde f_{0i} es el valor de f_i en X_0. X_0 es un punto en el espacio de soluciones formado por los valores x_{0j} que representan el punto de partida o solución inicial al comenzar el proceso de optimización del cual depende que tan buena será la solución final encontrada.

Desarrollando el binomio al cuadrado de la ecuación (3.5), tenemos

$$\phi = \sum_{i=1}^{\mathrm{M}} \mathrm{f}_{0i} + 2\sum_{i=1}^{M}\left[\mathrm{f}_{0i} \sum_{j=1}^{N} a_{ij}\left(x_j - x_{0j}\right)\right] + \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{N} a_{ij}\, a_{ik}\left(x_j - x_{0j}\right)\left(x_k - x_{0k}\right) \tag{3.7}$$

donde

$$a_{ij} = \frac{\partial f_i}{\partial x_j} \tag{3.8}$$

En la ecuación (3.7) el primer término es constante y puede despreciarse porque no influye en la topografía del espacio solución. El segundo y tercer termino puede combinarse cambiando el origen de x_j y haciendo una rotación de ejes de tal

forma que ϕ pueda expresarse como una forma cuadrática definida positiva [19] ($a_{ij}^2 \geq 0$ para toda i, j).

Para esto se deriva a ϕ con respecto a cada x_j, de la ecuación (3.7) y (3.8) se obtiene

$$\frac{\partial \phi}{\partial x_k} = \sum_{i=1}^{M} 2 f_i a_{ik} \text{ , para } k = 1,2,\ldots,N \tag{3.9}$$

sustituyendo f_i de la ecuación (3.6) e igualando con cero, se obtienen las siguientes N ecuaciones en $x_j - x_{0j}$

$$\sum_{i=1}^{M} f_{0i} a_{ik} + \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} a_{ik} \left(x_j - x_{0j} \right) = 0 \tag{3.10}$$

o en notación matricial,

$$A^T A\left(X - X_0\right) + A^T F_0 = 0 \tag{3.11}$$

donde A es una matriz de $M \times N$ con elementos a_{ij}, A^T es su transpuesta y F_0 es el valor de F_i evaluado en X_{0i}.

Las ecuaciones (3.10) o (3.11) son las ecuaciones clásicas de mínimos cuadrados, estas no necesitan que el número de variables sea igual al de ecuaciones. La solución de la ecuación (3.11) es

$$\mathrm{X} = -\left(\mathrm{A}^{\mathrm{T}}\mathrm{A}\right)^{-1}\mathrm{A}^{\mathrm{T}}\mathrm{F}_0 + \mathrm{X}_0 \tag{3.12}$$

la solución dada en (3.12) es un óptimo ya que el problema a resolver es lineal [16].

3.3 Solución del Problema

Como se mencionó en la sección 1.2, el primer paso para fabricar una superficie cónica consiste en generar una superficie esférica de referencia y a partir de ella, obtener la forma deseada. En la Figura 3.1 se puede observar que la parte sombreada es el material que se debe retirar para obtener el desgaste deseado.

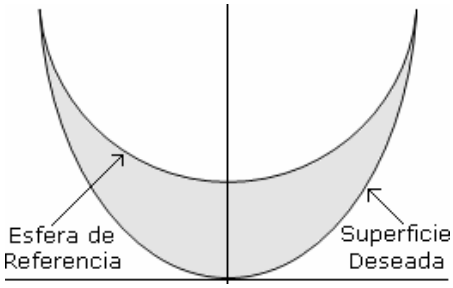


Figura 3.1 Generación de superficies cónicas a partir de una esfera de referencia.

Al conocer la superficie esférica que mejor se ajusta a la superficie cónica a fabricar, y medir sus diferencias se puede conocer los valores de la **función deseo**. Para ello se aplican diferentes criterios dependiendo de la forma de la superficie que se desee ya sea cóncava o convexa, ver Figura 3.2.

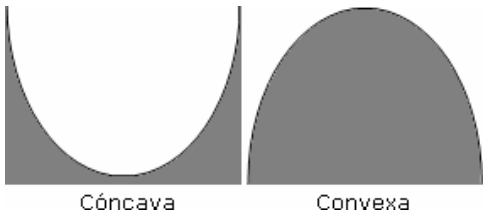


Figura 3.2 Tipo de superficies que se pueden generar en el Pulido Clásico.

Como se puede ver en la Figura 3.3, se observa que la distancia sagital o sagita en una superficie cónica en el eje de coordenadas (x , y , z), es la distancia que existe de un punto de la superficie cónica a un punto del eje de coordenadas.

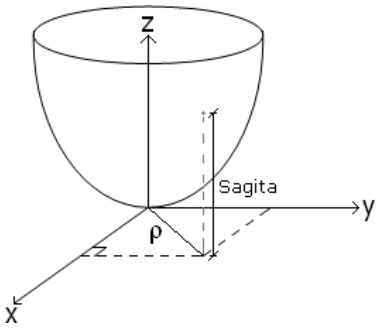


Figura 3.3 Superficie cónica vista desde el eje de coordenadas (x , y , z).

Si la superficie es cóncava, el criterio consiste en igualar pendientes en los bordes, por lo que

$$\left.\frac{d\ Z_e}{d\ \rho}\right|_{borde} = \left.\frac{d\ Z_c}{d\ \rho}\right|_{borde} \tag{3.13}$$

donde ρ es la distancia del centro de la superficie al punto de la superficie a evaluar, donde Z_e es la sagita de la esfera, Z_c es la sagita de la cónica, tal y como se visualiza en la Figura 3.4 y están definidas como

$$Z_e = \frac{C_e\ \rho^2}{1+\sqrt{1-C_e^2\ \rho^2}} \tag{3.14}$$

$$Z_c = \frac{C_c\ \rho^2}{1+\sqrt{1-(K+1)C_c^2\ \rho^2}} \tag{3.15}$$

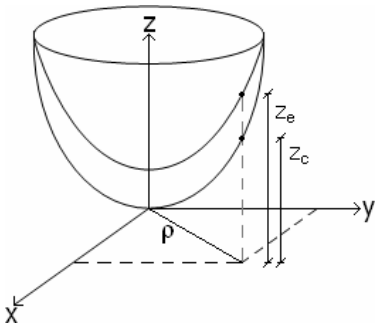


Figura 3.4 Superficie cónica y esfera de referencia vistas desde el eje de coordenadas (x , y , z).

donde C_e es la curvatura de la esfera, C_c es la curvatura de la cónica, K es la constante de conicidad, al derivar las expresiones (3.14) y (3.15) y resolviendo el sistema para C_e se obtiene

$$C_e = \frac{C_c}{\sqrt{1 - KC_c{}^2 \rho^2}} \tag{3.16}$$

que es la curvatura de la superficie esférica cóncava que mejor se ajusta a la superficie cónica deseada [2].

Ahora, si la superficie es convexa los criterios consisten en igualar las sagitas en el borde y en vértice, debiéndose cumplir

$$Z_e\big|_{borde} = Z_c\big|_{borde} \tag{3.17}$$

y

$$Z_e\big|_{vértice} = Z_c\big|_{vértice} \tag{3.18}$$

con Z_e y Z_c dados por (3.17) y (3.18), al despejar la curvatura de la esfera de referencia C_e queda como:

$$C_e = \frac{2C\left[1 + \sqrt{1 - (K+1)C_c^2 \rho_b^2}\right]}{\left[1 + \sqrt{1 - (K+1)C_c^2 \rho_b^2}\right]^2 + C^2 \rho_b^2} \tag{3.19}$$

Una vez que se conoce la curvatura de la esfera de referencia, el siguiente paso es conocer los valores de la función deseo en cada uno de los puntos a evaluar, para esto, se colocan las superficies (esférica de referencia y cónica a construir) con sus bordes coincidiendo, y midiendo la separación entre ellas, el desajuste punto a punto es mostrado en la Figura 3.5, y esta dado por

$$d = d_0 + \left[Z_e - Z_c\right] \tag{3.20}$$

donde d_0 es la separación en el origen entre la esfera de referencia y la cónica a fabricar.

$$d(\rho) = d_0 + \left[Z_e(\rho) - Z_c(\rho) \right] \tag{3.21}$$

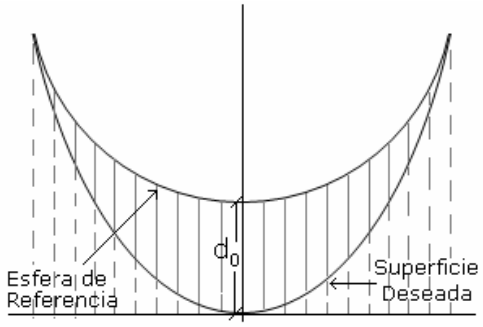


Figura 3.5 División de superficie en intervalos.

La superficie de vidrio se divide en intervalos, cada intervalo será trabajado por la herramienta sólida por determinado tiempo hasta obtener el desgaste deseado, ver Figura 3.6.

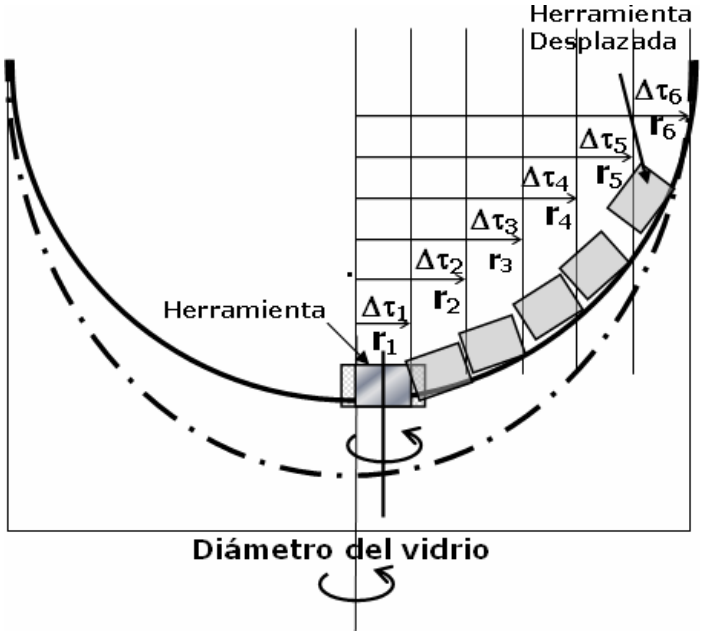


Figura 3.6 Proceso de pulido.

En cada posición de la herramienta se produce un desgaste h_{bi} definido por la ecuación (1.1) conocido como **función base**, calculado con el Modelo de Preston, ver Figura 3.7. Estas funciones están contenidas en archivos ya definidos y el simulador los une para formar una matriz.

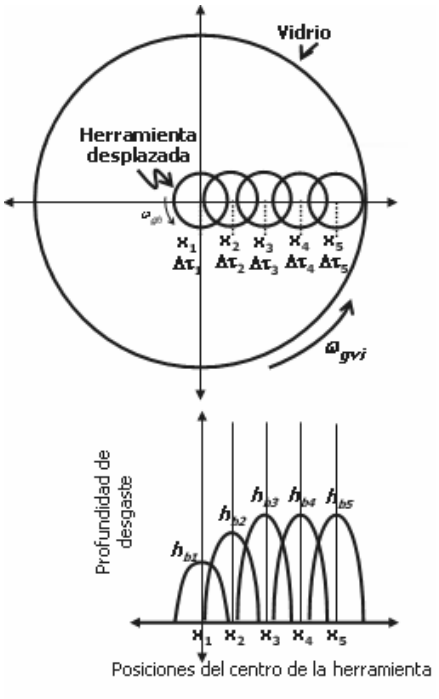


Figura 3.7 Funciones Base Generadas.

A partir de conocer las funciones base, y la superficie que se desea fabricar utilizando la técnica de pulido predecible, lo que se requiere es resolver el sistema definido por la ecuación (3.22).

$$d = \Delta t_1 b_1 + \Delta t_2 b_2 + \Delta t_2 b_2 + \Delta t_3 b_3 + \cdots + \Delta t_n b_n \tag{3.22}$$

donde d es la función deseo, b_i son las funciones base y Δt_i son los tiempos que la herramienta tiene que trabajar en las diferentes zonas de la esfera de referencia para generar la superficie cónica deseada.

En notación matricial, el problema a resolver esta dado como

$$Bt = d \tag{3.23}$$

o

$$\begin{bmatrix} b_{11}\ b_{12}\cdots & b_{1n} \\ b_{21}\ b_{22}\cdots & b_{2n} \\ .\ \ .\ \cdots & . \\ .\ \ .\ \cdots & . \\ .\ \ .\ \cdots & . \\ b_{m1}\ b_{m2}\cdots & b_{mn} \end{bmatrix} \begin{bmatrix} \Delta t_{11} \\ \Delta t_{21} \\ . \\ . \\ . \\ \Delta t_{n1} \end{bmatrix} = \begin{bmatrix} d_{11} \\ d_{21} \\ . \\ . \\ . \\ d_{m1} \end{bmatrix} \tag{3.23}$$

dado que el sistema no es una matriz cuadrada y que existen más vectores que variables a encontrar, el sistema está sobrevaluado por lo que existen más de una solución al mismo y el algoritmo a desarrollar deberá encontrar la solución que tenga las mínimas desviaciones, las cuales serán dadas por la función de evaluación, ver ecuación (3.24).

$$\varepsilon_i = \left[\sum_{i=1}^{N} \Delta t_i\, b_i - d_i \right]^2 \tag{3.24}$$

Conocida la función deseo, se resuelve el sistema por medio de mínimos cuadrados. Al derivar la ecuación (3.23) con respecto a Δt_i e igualar a cero, se obtiene

$$\sum_{i=1}^{N} \left(\sum \Delta t_i\, b_{ij} - d_i \right) b_{ik} = 0 \tag{3.25}$$

reagrupando y despejando los términos de la función deseo, da

$$\sum_{j=1}^{N} \Delta t_j \sum_{i=1}^{N} b_{ij}\, b_{ik} = \sum_{i=1}^{N} d_i\, b_{ik} \tag{3.26}$$

ahora, despejando Δt y escribiendo en notación matricial [16]

$$\Delta t = \left[B^T B \right]^{-1} B^T D \tag{3.27}$$

Finalmente, el software resuelve el sistema que define la ecuación (3.27) para obtener los tiempos de estancia de la herramienta. Para ilustrar el funcionamiento del software se presenta el diagrama de flujo en la Figura 3.8.

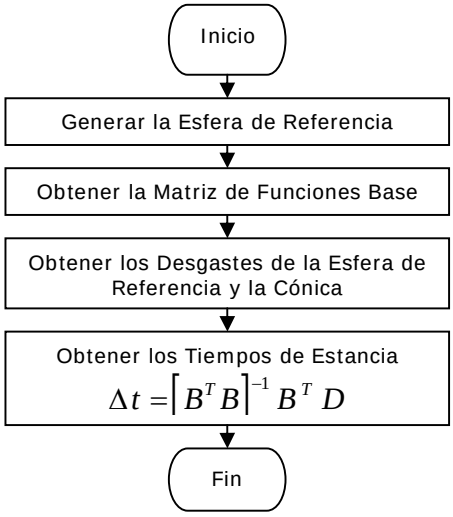


Figura 3.8 Diagrama de Flujo de Mínimos Cuadrados.

Cabe mencionar, que en el capítulo 6 se describen los resultados obtenidos con el método de mínimos cuadrados aplicado a la solución del problema mencionado a lo largo de este capítulo.

Capítulo 4. **Aplicación de Algoritmos Genéticos al Pulido de Superficies**

4.1 Introducción

En la actualidad, en el pulido de superficies con el método clásico, al fabricar una superficie, se genera una esfera inicial y a partir de ella, con una herramienta de 1/5 el tamaño de la superficie de vidrio con la que se está trabajando, se busca generar el perfil deseado. Al trabajar con una herramienta de pétalo, cuando se requiera generar una superficie, se pretende que en lugar de dividir la superficie en intervalos, se trabaje con una herramienta, aproximadamente del 90% del tamaño del vidrio, que oscile y gire para generar el desgaste deseado, el único inconveniente es que el técnico deberá supervisar en todo momento el proceso para no desgastar el vidrio más de lo que se requiere [17, 18]. Para diseñar una herramienta de este tipo se pretende obtener los tamaños angulares (ver sección 1.1) óptimos de los anillos concéntricos que formen el perfil de la herramienta de tal manera que genere el desgaste deseado en una superficie de vidrio. En este capítulo se describe el método de Algoritmos Genéticos y como fue implementado para resolver el problema de encontrar los tamaños angulares.

4.2 Algoritmos Genéticos (AG)

En general, las técnicas de optimización de Algoritmos Genéticos, consisten en hallar el extremo (un máximo o un mínimo) de una función de aptitud convenientemente elegida [19], Los AG están basados en el mecanismo de selección natural y genética natural, es decir, utilizan una analogía directa del fenómeno evolutivo en la naturaleza para resolver problemas. El tema principal de la búsqueda que se realiza con algoritmos genéticos ha sido la robustez, el balance entre la eficiencia y la eficacia necesaria para la supervivencia en diversos entornos [20].

Los algoritmos genéticos pueden ser utilizados en la resolución de problemas siempre y cuando, el problema se pueda codificar en un cromosoma y que se pueda asociar a una función de evaluación o función de aptitud que dé una medida de valor de los cromosomas en el contexto del problema.

Los Algoritmos Genéticos se diferencian de los demás procedimientos de búsqueda normales en lo siguiente:

- Los AG trabajan con un código de parámetros y no con los parámetros mismos.
- Los AG buscan poblaciones de puntos y no puntos individuales.
- Los AG usan la función de aptitud o de mérito y no sus derivadas u otros conocimientos auxiliares.
- Finalmente, los AG usan reglas de transición probabilística, no reglas deterministas.

La estructura de un algoritmo genético simple se compone de un generador de números aleatorios, módulo de evaluación, función de aptitud, módulo de selección, de mutación y de cruza [21].

En un AG, los individuos pueden ser representados de diversas maneras, de forma binaria (unos y ceros), por un número real entre otras, así también contiene diversos parámetros tales como longitud del cromosoma, tamaño de la población, número máximo de generaciones, probabilidad de cruza y probabilidad de mutación, los cuales son seleccionados de acuerdo al problema con el que se este trabajando.

A continuación, se presenta el comportamiento de cada uno de los módulos que constituyen a un algoritmo genético simple.

4.2.1 Generador de Números Aleatorios

Para trabajar con algoritmos genéticos se debe tener una población inicial representada por un conjunto de cromosomas, a los que se aplicarán todos los mecanismos de selección natural. La población inicial es generada de manera aleatoria, de acuerdo a la representación elegida, se crea la población generado aleatorios.

4.2.2 Módulo de Evaluación

Obtenida la población inicial, se hace uso de una función de aptitud, esta función depende de cada problema en particular y dicha función define los resultados del algoritmo genético. Durante la evaluación se decodifica cada cromosoma, convirtiéndose cada uno de ellos en una posible solución del problema. A cada posible solución (dada por cada cromosoma) se le da una puntuación a esa solución en función de lo cerca que esté de la mejor solución. A esta puntuación se le llama fitness o aptitud [21].

4.2.3 Selección de Cromosomas

Una vez conocida la aptitud de cada cromosoma, se seleccionan los cromosomas que se van a reproducir y posteriormente a cruzar para dar lugar a una nueva generación. En la selección, tentativamente, los cromosomas mejor evaluados tienen mayor posibilidad de ser escogidos. La finalidad de seleccionar a los cromosomas, es para darles más oportunidades de reproducirse a los miembros de la población que son más aptos o que tiene mejores evaluaciones. Existen diversas técnicas de selección, tales como los que se muestran en la Figura 4.1.

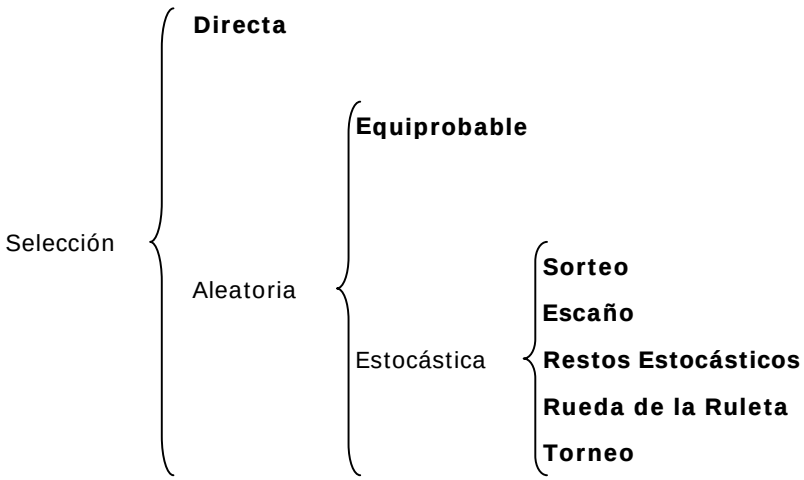


Figura 4.1 Tipos de selección [21].

Después de seleccionar a los cromosomas, se debe dar paso a una nueva generación, es decir, los individuos o cromosomas de la población deben reproducirse.

4.2.4 Cruza de Cromosomas

En un algoritmo genético una cruza recombina o intercambia el material genético de dos cromosomas para crear dos hijos o descendientes. Existen diversas formas de llevar a cabo el intercambio genético, pero hay dos grupos principales, cruza en n puntos (los dos cromosomas se cortan en n puntos y se intercambia el material genético de dichos cromosomas, los más comunes son el de uno y dos puntos) y cruza uniforme (se genera un patrón aleatorio de unos y ceros, y se intercambian los bits de los dos cromosomas que coincidan donde hay un 1 en el patrón o se genera un número aleatorio para cada bit, y si supera una determinada probabilidad se intercambia ese bit entre los dos cromosomas) [21]. La cruza es un componente importante en un algoritmo genético, cabe mencionar que se le da una probabilidad con la que la cruza se llevará a cabo que determinará que no siempre se reproducirán los cromosomas padres.

4.2.5 Mutación

El proceso de mutación consiste en seleccionar aleatoriamente un cromosoma de la población y mutarlo, para ello existen diversas técnicas, entre las que se encuentran la mutación de bit, multibit, de gen, multigen, de intercambio, de barajado entre otras [22]. El proceso de mutación comúnmente utilizado en algoritmos genéticos simples es la mutación de bit, en la cual se selecciona aleatoriamente un bit del cromosoma y se cambia el valor de ese bit de uno a cero o viceversa. Una mutación impide que las poblaciones se vuelvan homogéneas y mantiene la diversidad, permitiendo así que el proceso de evolución continúe avanzando.

En la siguiente sección, se describe detalladamente el problema que se presenta en el pulido clásico y la solución del mismo por medio de algoritmos genéticos.

4.3 Solución del Problema

Como se mencionó al inicio de este capítulo, en la actualidad, se pretende realizar el proceso de pulido clásico por medio de una herramienta de pétalo con un tamaño similar a la superficie de vidrio a pulir, este cambio en el proceso generará

algunas ventajas tales como que el técnico no tendrá que detener el proceso de pulido para mover la herramienta a otra zona de la superficie de vidrio así como también, tendrá la seguridad de obtener el desgaste que desea de la superficie con la herramienta de pétalo, sin embargo, él deberá estar al pendiente en todo momento del proceso.

Una herramienta de pétalo esta formada por 100 anillos incompletos, los tamaños angulares de los anillos forman los pétalos que constituyen a esta herramienta, de acuerdo a su forma, se pueden generar diversos desgastes, de ahí la importancia de optimizar dichos tamaños angulares. La Figura 4.2 muestra una herramienta que por tener los pétalos muy gruesos realiza un desgaste distinto al que se requiere; como se puede observar en la parte derecha, el desgaste que produce esta herramienta (desgaste simulado) es muy diferente al que se desea obtener (desgaste deseado), por lo cual no es apropiado trabajar con esa herramienta, ya que no se obtendrá el perfil deseado.

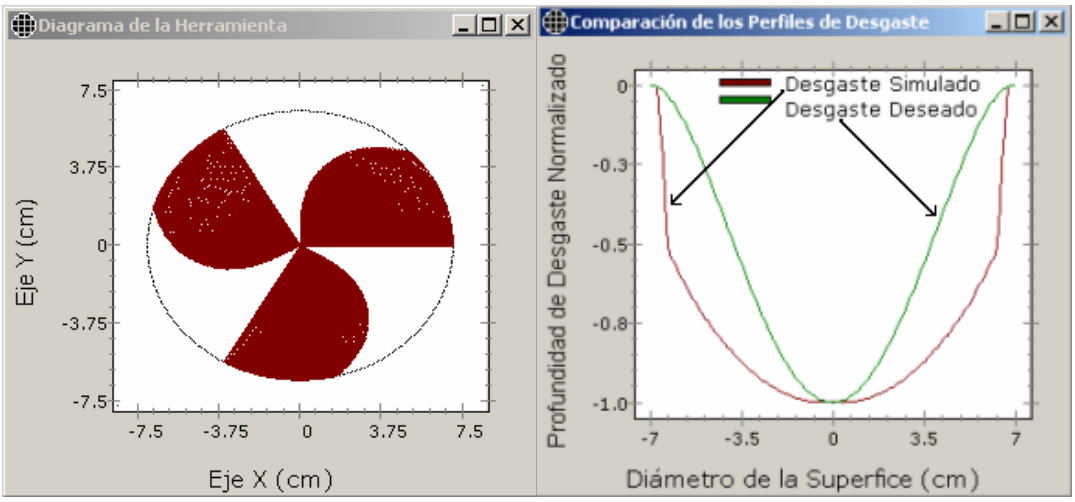


Figura 4.2 Herramienta de pétalo no optimizada.

No obstante, si se utiliza una herramienta con el ancho de los pétalos apropiados, se puede obtener el desgaste deseado tal y como lo muestra la Figura 4.3, en la que se puede visualizar que el desgaste generado o simulado por la herramienta es muy similar al deseado.

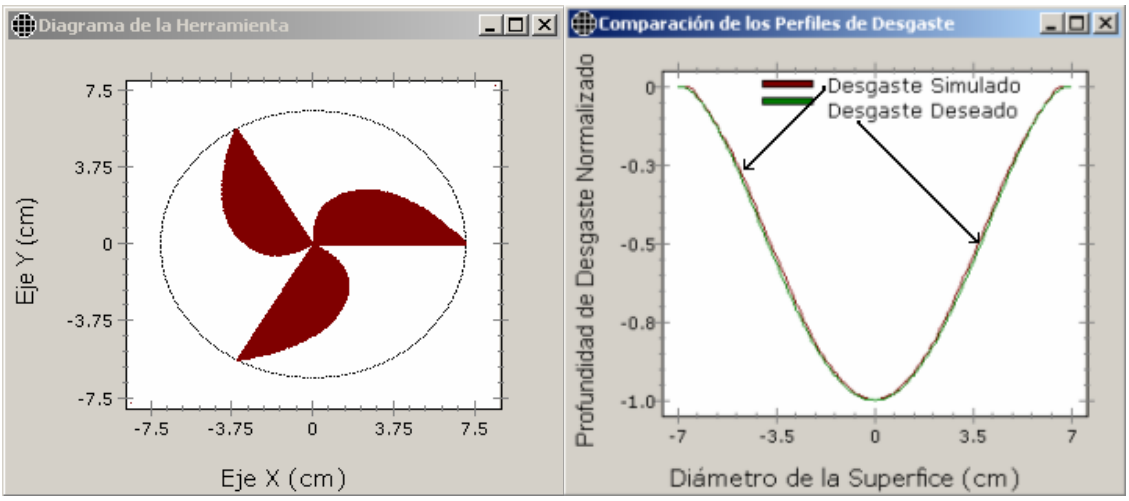


Figura 4.3 Herramienta de pétalo optimizada.

Para optimizar la forma de los pétalos de la herramienta, se deben obtener los tamaños angulares de los anillos, pero como se mencionó al inicio de esta sección, la herramienta esta constituida por 100 anillos, esto hace que sean una gran cantidad de valores a optimizar. Para obtener los tamaños angulares, se hace uso de variables llamadas incrementos, cada incremento cubre a 10 anillos tal y como lo muestra la Tabla 4.1, en la cual se tienen cada uno de los incrementos representados por m_j y los anillos que cubre cada incremento representados por α_i [18].

Tabla 4.1 Incrementos para obtener los anillos de la Herramienta de pétalo.

# de Incremento	Incrementos (m_j)	Anillos (α_i)
1	m_1	$\alpha_{i+1} = \alpha_i + m_1,\ i = 1,2,...,10$
2	m_2	$\alpha_{i+1} = \alpha_i + m_2,\ i = 11,12,...,20$
	. . .	
10	m_{10}	$\alpha_{i+1} = \alpha_i + m_{10},\ i = 91,92,...,100$

Para obtener los valores optimizados de los 10 incrementos que formarán los anillos de la herramienta de pétalo, se aplicó algoritmos genéticos. Cabe mencionar que se optimiza uno a uno cada incremento por lo que se repite el AG 10 veces, tal y como se observa en el diagrama de flujo de la Figura 4.4.

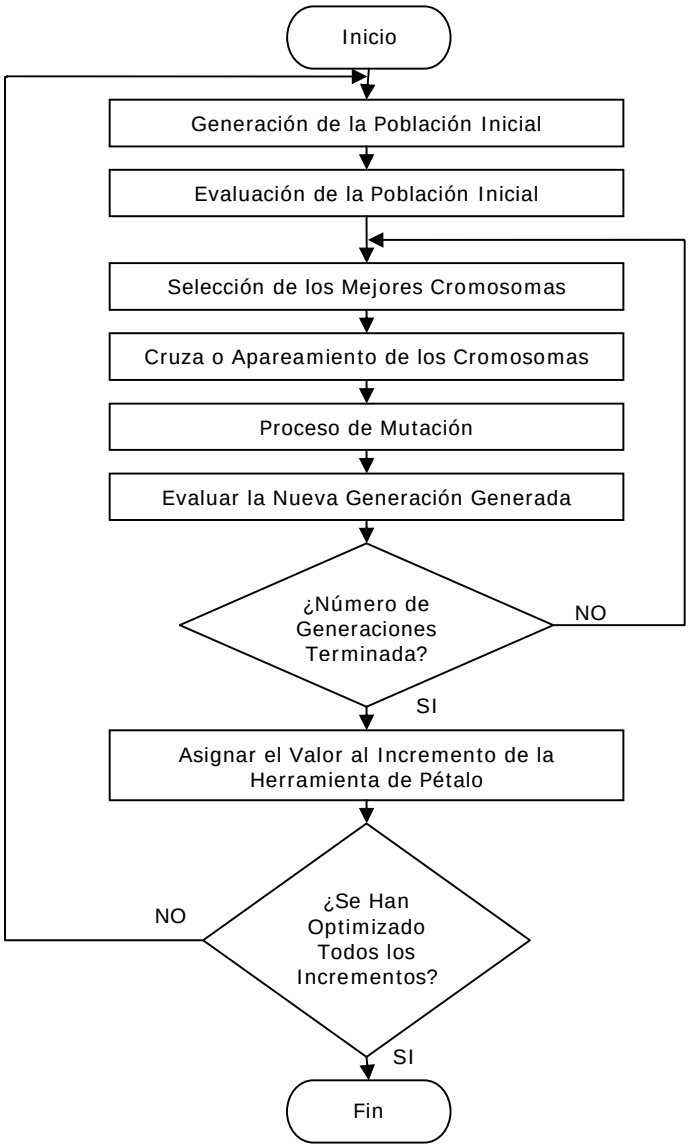


Figura 4.4 Diagrama de Flujo del Algoritmo Genético.

En seguida, se describe cada uno de los bloques mostrados en el diagrama de flujo, iniciando con la generación de la población inicial.

4.3.1 Generación de la población inicial de cromosomas

Para generar la población inicial, al aplicar el algoritmo al problema del SSPC, se generaron cadenas de 8 bits formada por unos y ceros (ver Figura 4.5). Cada cadena simboliza a un cromosoma y a su vez un valor generado para un determinado incremento m_j. El número de bits utilizado es de 10 (ver sección 6.2.3).

1	0	0	1	1	0	1	1	0	1

Figura 4.5 Cromosoma.

Cada cromosoma contiene un valor del incremento que se esté optimizando, ya que esta codificado en binario de ocho bits, representa a un número en decimal que va de 0 a 255. Los valores óptimos de los incrementos pueden ser menores o mayores a cero y están comprendidos en el rango de (-5, 5), este rango fue seleccionado al trabajar con el método de prueba y error (para optimizar los tamaños angulares), ya que por medio de él se observó que los valores de los incrementos se encontraban en este rango; por lo cual el número decimal debe ser transformado a ese rango, esto se puede ver en la Tabla 4.2, en la cual se tiene un cromosoma en codificación binaria y su equivalente en decimal, el cual es transformado en un número que se encuentra en el rango.

Tabla 4.2 Decodificación del Cromosoma.

Valor Binario	Valor Decimal	Valor en el Rango
1001 101111	623	1.083984375

Esta decodificación se hará para cada uno de los cromosomas que forman la población con un tamaño de 20 individuos; este tamaño de población se seleccionó por dos razones: 1) para reducir al mínimo el tiempo de cómputo (en este caso de 10 minutos aproximadamente) y 2) con este tamaño de población se obtiene un resultado satisfactorio para este proyecto de tesis. Posteriormente, se describe la evaluación realizada a dicha población.

4.3.2 Evaluación de la Población

Para saber qué cromosoma se acerca al valor óptimo del incremento que se está optimizando, se le asigna una puntuación de acuerdo a lo cerca que se encuentre de la mejor solución, para ello, se utilizó la función de aptitud que se describe a continuación.

Función de Aptitud

La función de aptitud utilizada en la solución de este problema, consiste en restar los desgastes simulados a los experimentales o deseados en cada punto sobre el vidrio, tal y como se muestra en la Figura 4.6, para ello, de acuerdo a los parámetros introducidos por el usuario, el software de simulación obtiene el valor de estos desgastes aplicando la ecuación de Preston mencionada la sección 1.3.1.

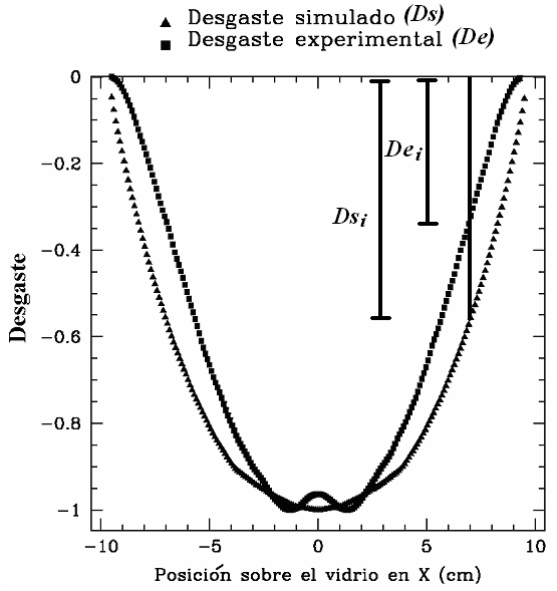


Figura 4.6 Resta de los desgastes experimental y simulado para cada punto sobre el vidrio.

El objetivo de restar estos desgastes, es saber que tan cerca se encuentra el perfil simulado del que se desea obtener, cuando el valor obtenido se acerca más a cero, significa que se está obteniendo un desgaste similar o cercano al deseado, entre más alejado este del cero más lejos se estará del desgaste deseado. Finalmente, la diferencia de desgastes se eleva al cuadrado con lo que se obtienen valores positivos. Por lo tanto, es la ecuación (4.1), la que funge como función de aptitud.

$$Suma = \sum_{i=-n}^{n} Error_i \qquad (4.1)$$

donde:

$Error_i =$ Es la diferencia entre el desgaste simulado y el desgaste experimental al cuadrado (ecuación 4.2).

$n =$ Número de puntos en los cuales se calcula el desgaste sobre el vidrio.

Esto es:

$$Error_i = \left(Ds_i - De_i\right)^2 \tag{4.2}$$

en donde Ds_i es el desgaste simulado generado por la herramienta y De_i es el desgaste deseado o experimental.

El comportamiento de la ecuación (4.1), proporciona un mínimo; como los algoritmos genéticos trabajan con máximos, a esta ecuación se multiplica por menos uno y se le suma una constante C, con el objetivo de que ahora se tenga una función de aptitud que proporcione un máximo cuyo valor sea la constante C. El valor de la constante C utilizada, fue $C = 100$ para garantizar evaluaciones positivas al aplicar la etapa de selección (ver sección 4.3.3), es decir, el cromosoma cuyo valor en decimal produce la evaluación más cercana a 100, será considerado el mejor valor de la variable. Así, se evalúa cada uno de los cromosomas de la población, en la Tabla 4.3 se muestra la puntuación obtenida por el cromosoma 1.05529296875.

Tabla 4.3 Evaluación de los cromosomas.

Valor del Cromosoma	Evaluación
1.05529296875	99.8890406140753

El siguiente paso del algoritmo genético, es realizar el proceso de selección tal y como se presenta en la siguiente sección.

4.3.3 Selección de Cromosomas

Para seleccionar a los cromosomas en este AG, se utilizó la técnica de selección por rueda de la ruleta, ya que es la técnica más común aplicada a los algoritmos genéticos simples, la cual consiste en tomar a la población como un pastel, en el cual

cada cromosoma le corresponde una rebanada en proporción a su aptitud, finalmente se seleccionan los cromosomas aleatoriamente y se espera que los mejores cromosomas sean los que tengan más probabilidad de ser seleccionados. En el caso del SSPC se hace una suma acumulada de las evaluaciones, después se selecciona aleatoriamente un número que sea menor o igual a esta suma, se va sumando uno a uno la evaluación de cada cromosoma y se selecciona al que sea mayor o igual a este número. Por ejemplo, supongamos que se tiene una población de cinco cromosomas y cada cromosoma tiene una evaluación a la cual se hace la suma acumulada (ver Tabla 4.4).

Tabla 4.4 Ejemplo de Selección

Cromosoma	Evaluación	Suma Acumulada
0.380706787109375	99.9948357941699	99.9948357941699
-0.469818115234375	99.9984570639738	199.993292858144
-1.6119384765625	99.9531853182459	299.94647817639
1.36322021484375	99.9391949514842	399.885673127874
-2.62405395507812	99.8560756552221	499.741748783096

En base al resultado de la suma acumulada (499.741748783096), se crea un número aleatorio que comprenda el rango de (0, 499.741748783096) y se selecciona el cromosoma que sea mayor o igual a ese número, si el resultado del aleatorio fuera 252 el cromosoma seleccionado sería -1.6119384765625. Este proceso se repite el número de veces del tamaño de población, en el caso del SSPC es 20.

Posteriormente, se realiza el proceso de cruza como se muestra a continuación.

4.3.4 Cruza de Cromosomas

Como se mencionó en la sección 4.2.4 se tienen diferentes técnicas de cruza de cromosomas, en la solución de este problema se utilizó la cruza en un punto, nuevamente por ser la más sencilla de implementar y más utilizada en los AG simples, este tipo de cruza consiste en recombinar el material genético de los cromosomas seleccionados, se toman dos cromosomas que se cortan en un punto, generado aleatoriamente, y el material genético situado entre ellos se intercambia.

En el caso de este AG, primero se toman los padres de la lista de cromosomas seleccionados, se genera un número aleatorio entre uno y el tamaño máximo de la longitud del cromosoma (en este caso ocho), después se hace un corte en el punto obtenido del aleatorio y cruzan dichos cromosomas, en la Figura 4.7 se puede observar este proceso, se generó un número aleatorio (en este caso tres), y se hace el corte en ese punto, después se combinan los cromosomas copiando tal cual los tres primeros bits y cambiando entre si, los últimos bits generando así nuevos cromosomas (hijos).

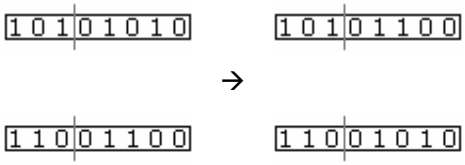


Figura 4.7 Cruza de dos cromosomas.

Después, se mutan los cromosomas tal y como se describe en la siguiente sección.

4.3.5 Mutación de un Cromosoma

La técnica uniforme de mutación utilizada en el AG para la solución de este problema fue la de un bit, debido que es la técnica más sencilla y más utilizada, en la cual se selecciona aleatoriamente tanto el cromosoma como el bit que será mutado, y se realiza la operación de mutación, la cual consiste en cambiar el valor del bit, por ejemplo si en la casilla se encuentra un 0, se cambia a 1 o viceversa si hay un 1 se cambia por un 0, ver Figura 4.8.

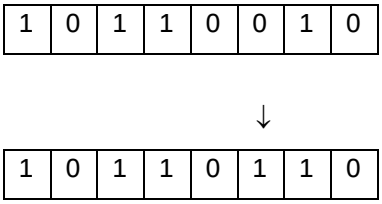


Figura 4.8 Mutación en un cromosoma.

Al terminar la mutación, se genera una nueva población, la cual se evalúa y se obtiene el mejor individuo, si se realizaron el número de generaciones indicadas se asigna el mejor cromosoma al incremento que se este optimizando, sino, se realizan nuevamente los procesos de selección, cruza, mutación y evaluación. El algoritmo

genético se ejecuta 10 veces ya que son el número de incrementos que se desea optimizar. Los resultados obtenidos son mostrados en el capítulo 6 de este documento.

Una forma de cuantificar la diferencia que existe entre el desgaste generado por la herramienta de pétalo (calculada con AG) y el desgaste deseado, es utilizando la Raíz Media Cuadrática (RMS) cuyas siglas provienen del inglés Root Mean Sqare. El RMS es calculado como sigue

$$RMS = \sqrt{\frac{Error^2}{N-1}} \tag{4.3}$$

en donde N es el número de puntos que se tomaron en cuenta para el cálculo del desgaste, en este caso 101 puntos. En cuanto más pequeño sea el valor RMS, más parecido es el desgaste generado con el deseado.

Finalmente, tanto los parámetros como los resultados obtenidos con el algoritmo genético se describen en el capítulo 6.

Capítulo 5. **Pruebas de Usabilidad Aplicadas al Software de Pulido**

5.1 Introducción

Durante el proceso de desarrollo de un software, se espera cumplir con las expectativas del usuario al fijar diferentes objetivos, entre los cuales tenemos que el software obtenga los resultados esperados y que funcione con la menor cantidad de errores. Además, muchas veces no se toma en cuenta que en la implementación y distribución del software, el usuario final es el que decidirá su grado de utilidad; por tal motivo, el desarrollo de la **interfaz** es de vital importancia, debido a que puede determinar el éxito o fracaso del software, este aspecto genera la necesidad de realizar pruebas de usabilidad.

La usabilidad significa que la gente quien usa el producto puede llevar a cabo rápida y fácilmente sus propias tareas y se resume en cuatro puntos [23]:

- Usabilidad significa enfocarse en los usuarios.
- La gente usa el producto para ser productivo.
- Los usuarios son personas intentando llevar a cabo las tareas.
- El usuario decide cuando un producto es fácil de usar.

La Organización Internacional para la Estandarización (ISO) en la norma ISO/IEC 9124 nos dice que “la usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso” [24].

Finalmente, las pruebas de usabilidad tienen cinco características importantes [23]:

- El objetivo primario es mejorar la usabilidad del producto
- Los participantes representan usuarios reales.
- Los participantes realizan tareas reales.
- El observador graba lo que los participantes hacen y dicen
- Se analizan los datos, se diagnostican los problemas y se recomiendan cambios para solucionar estos problemas.

En las siguientes secciones, se describen las pruebas de usabilidad realizadas al SSPC.

5.2 Problema a Resolver

Que el simulador sea entendible y usable para los técnicos que laboran en el taller de pulido, ya que serán beneficiados por dicho software al servirles de consulta y soporte.

5.3 Solución del Problema

Para verificar la que tan usable es el SSPC, se le aplicaron pruebas de usabilidad; dichas pruebas harán que los usuarios obtengan una mayor facilidad de **uso**, **flexibilidad** y **robustez** del mismo, así como también las ventajas que se describen en el siguiente apartado.

Ventajas de la Usabilidad

Al tomar en cuenta al usuario en todo momento en el desarrollo de cualquier sistema y efectuar pruebas de usabilidad se obtienen los siguientes beneficios [24]:

- Reducción de costos de producción. Gracias a que se aplicaron éstas pruebas, los errores fueron encontrados a tiempo y no cuando el programa esté en manos de los usuarios.

- Reducción de costos de aprendizaje. Debido a que el simulador es más fácil de entender, se requiere de menos entrenamiento y asistencia al usuario, por lo que reduce costos de soporte a usuarios.
- Aumento en la productividad. Ya que el simulador se ajusta a las necesidades de los usuarios, incrementa la satisfacción de uso y se convierte en un aliado en la producción de superficies ópticas.
- Mejora la calidad del producto. Al aplicar las pruebas al simulador, se detectaron los errores y se mejoró el funcionamiento del programa, esto le proporciona una mejor calidad de uso por lo que se vuelve más eficiente.

5.4 Pruebas Realizadas

Las pruebas que se realizaron al simulador tuvieron el objetivo de mejorar la interfaz de usuario, es decir, que los elementos que interactúan con el usuario fueran lo suficientemente entendibles y que los resultados obtenidos por el software sean los esperados por los usuarios. Como se mencionó en la sección 2.2.3, se realizaron las pruebas en dos lugares diferentes, las primeras pruebas fueron hechas en el UsaLab. Posteriormente, se realizaron pruebas en el taller de pulido del INAOE.

La regla básica para realizar las tareas de las pruebas de usabilidad es que deben ser elegidas de tal manera que se acerquen a los usos que tendrá el software; estas pueden ser diseñadas en base a un análisis o basadas en una lista de usos previstos para el producto [25]. En el caso del SSPC se realizaron ocho tareas (Tabla 5.1), elegidas de acuerdo a las necesidades del proyecto.

En el desarrollo de las pruebas se tuvieron tres usuarios en la UTM (usuarios 1, 2 y 3) y siete usuarios en el INAOE (usuarios 4, 5, 6, 7, 8, 9 y 10), cada uno de los usuarios realizaron al menos una de las diferentes tareas mostradas en la Tabla 5.1, de las cuales se obtuvieron las modificaciones generales presentadas en la Figura 5.1.

Tabla 5.1 Lista de tareas.

Prueba	Descripción
1	El usuario deberá hacer uso de la herramienta sólida.
2	El usuario hace uso de una herramienta anular.
3	El usuario utiliza una herramienta de Pétalo para generar una superficie cóncava.
4	El usuario utiliza una herramienta de Pétalo para generar una superficie convexa.
5	El usuario deberá iniciar con la opción tiempos de estancia de la herramienta sólida para una superficie cóncava.
6	El usuario deberá iniciar con la opción tiempos de estancia de la herramienta sólida para una superficie convexa.
7	El usuario deberá iniciar la optimización de la herramienta de pétalo para una superficie cóncava con archivo de datos.
8	El usuario deberá iniciar la optimización de la herramienta de pétalo para una superficie cóncava sin archivo de datos.

En la Figura 5.1, se puede observar que el usuario 1 sugirió modificar las unidades de los parámetros: ayuda del simulador, lenguaje, interfaz e idioma.

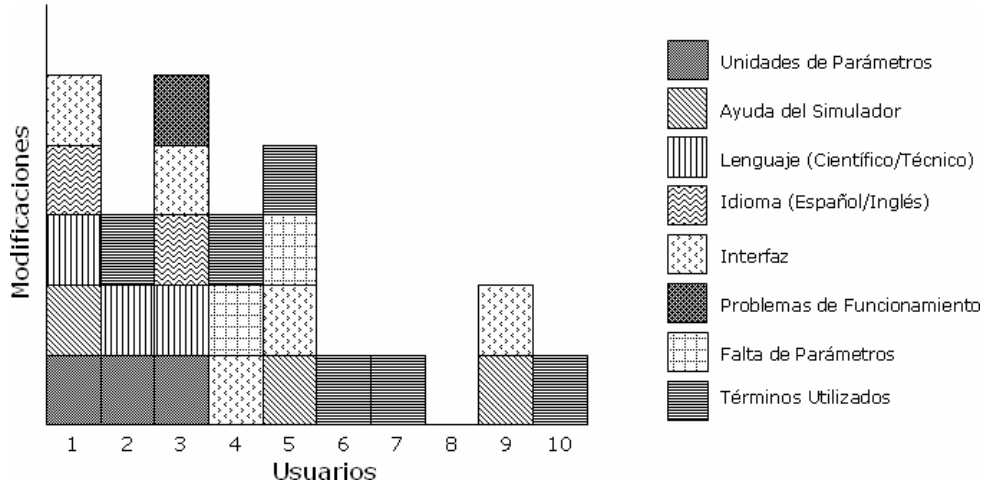


Figura 5.1 Resultados obtenidos en las pruebas de usabilidad.

En la Figura 5.2 se muestra la gráfica correspondiente a la satisfacción de los usuarios al utilizar el simulador de pulido, en la cual se observa que en promedio no

estuvieron satisfechos en su totalidad debido a los problemas presentados en la sección 2.4.

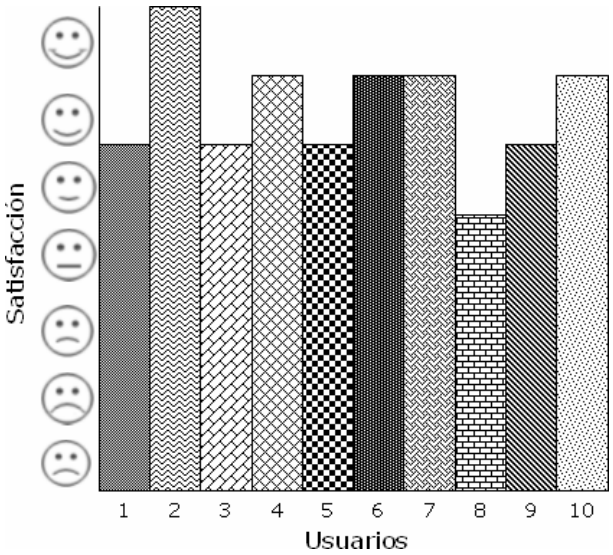


Figura 5.2 Satisfacción de los usuarios al utilizar el simulador.

Dadas las modificaciones sugeridas, la siguiente sección muestra los cambios realizados al SSPC.

5.5. Modificaciones Generales

Durante el desarrollo de las pruebas hubo algunos errores de funcionamiento del software que fueron detectados y luego modificados en versiones posteriores. Las modificaciones que se realizaron en el software son las siguientes.

UTM

- El usuario podrá elegir el idioma del simulador, Español o Inglés.
- Se cambiaron las unidades de alguno de los parámetros.
- Cambiar el término utilizado en la entrada de datos de los anillos concéntricos que forman a la herramienta de pétalo.
- La pantalla de presentación se mantenía por un minuto, lo cual se cambió a una pantalla que dura el tiempo que el usuario lo requiere, es decir, le dará un clic para continuar con la pantalla principal.

INAOE

- Se cambió el término de algunos parámetros como:
  - Excentricidad por Amplitud de Oscilación (Carrera).
  - Ángulo de descentramiento por posición de la herramienta en el vidrio.
- Se cambió la etiqueta del botón "Aceptar" en la modificación de los parámetros por "Iniciar Simulación".
- Se cambió el tamaño y el tipo de letra para que sea más visible para el usuario.

A continuación, se presentan las figuras con las modificaciones hechas al software de simulación (Figura 5.3 a 5.7). Inicialmente, el SSPC fue realizado en Idioma Inglés, dado que los técnicos no tienen conocimientos en este idioma, surgió la necesidad de que el SSPC soportará los idiomas Español e Inglés.

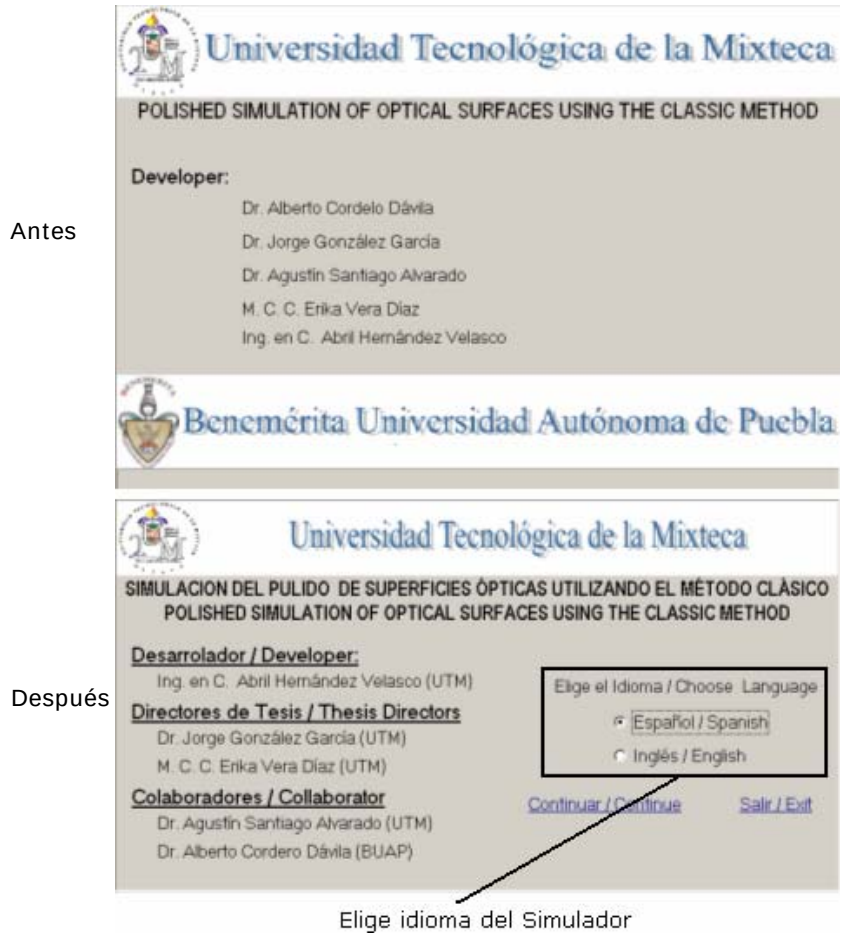
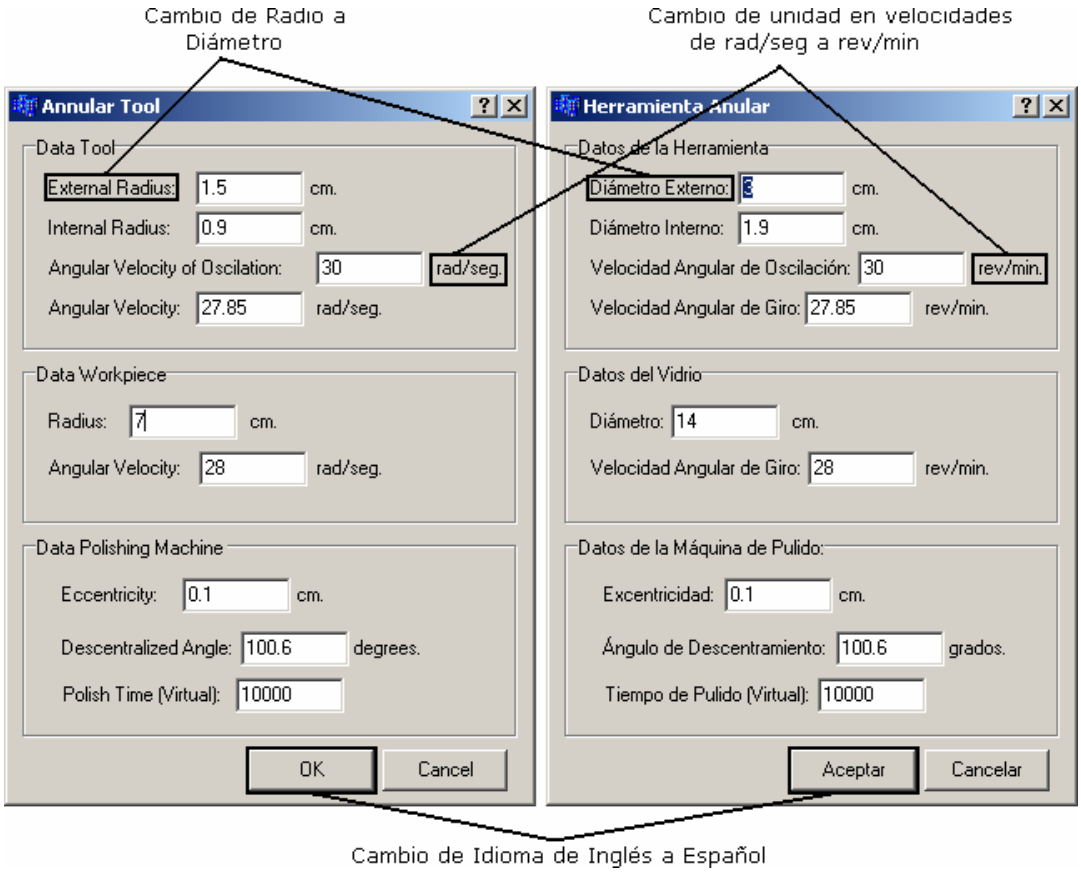


Figura 5.3 Elección de idiomas.



(a) (b)

Figura 5.4 Cambio de unidades de los parámetros de pulido. (a) Versión inicial, (b) Versión final.

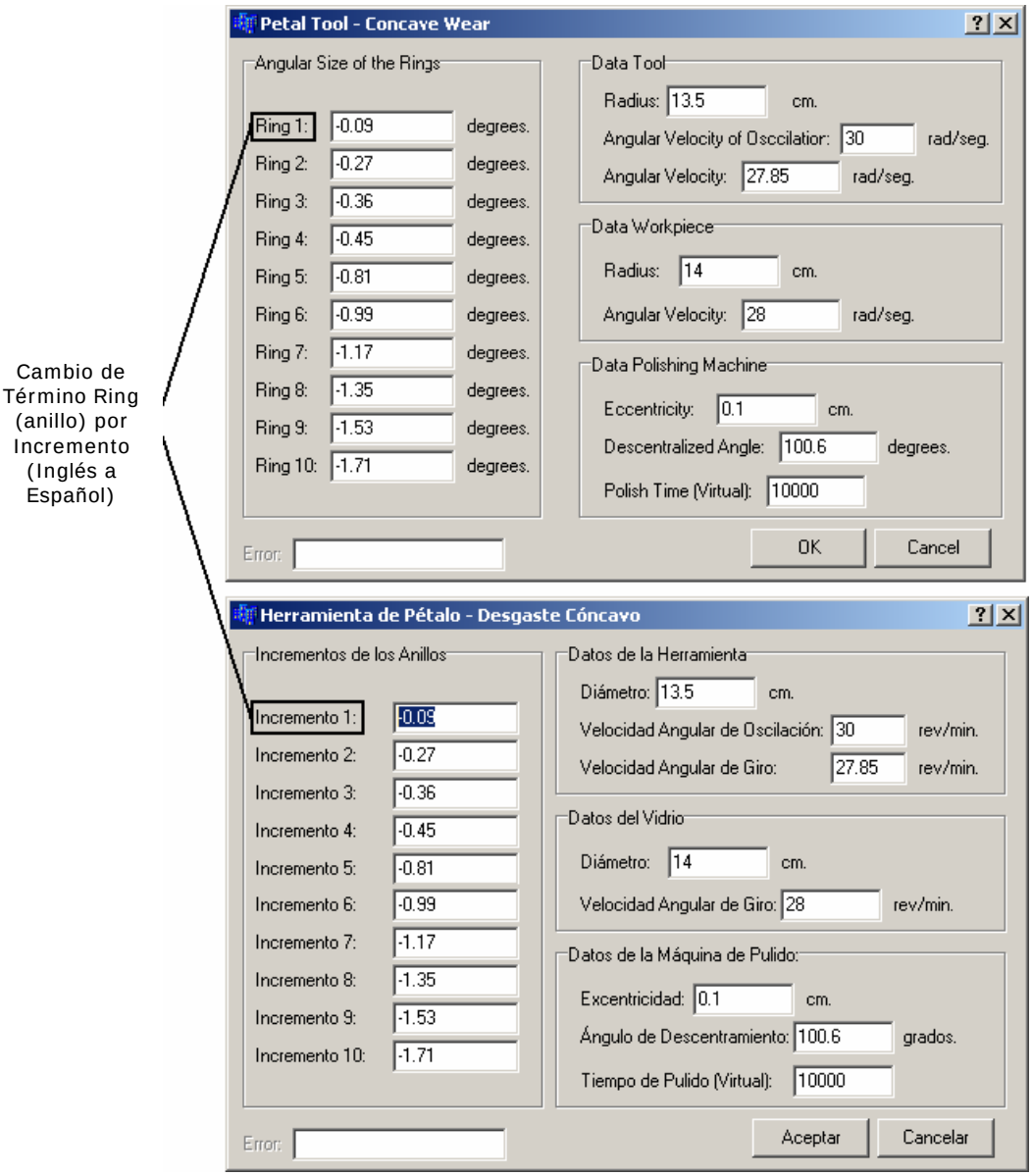
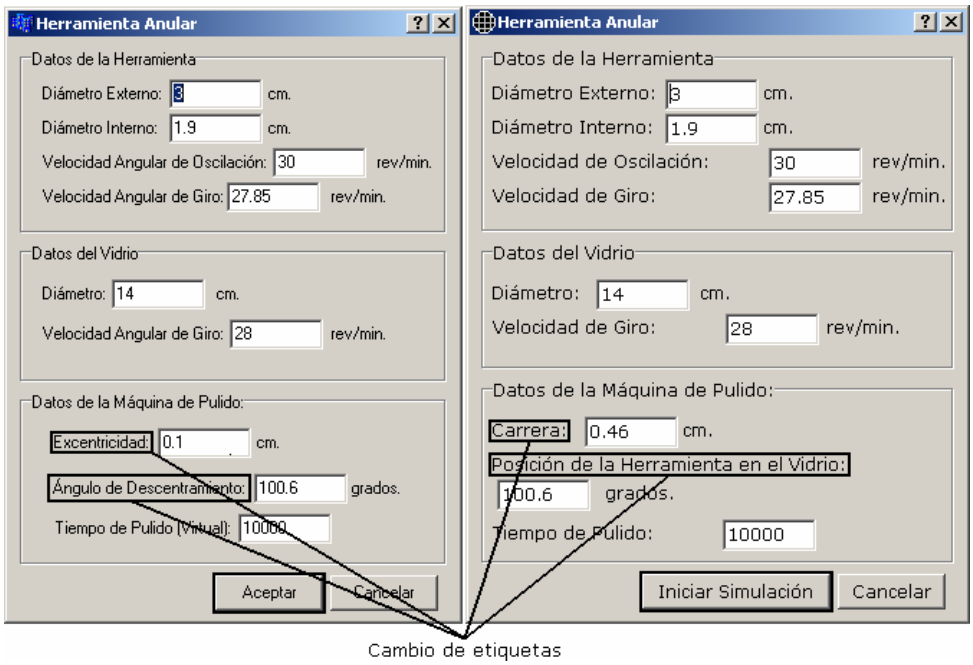


Figura 5.5 Términos en los anillos de la herramienta de pétalo.

(a) (b)

Figura 5.6 Cambio del tamaño de letra en el simulador. (a) Versión inicial, (b) Versión final.



(a) (b)

Figura 5.7 Cambio de términos de parámetros y etiquetas. (a) Versión inicial, (b) Versión final.

Finalmente, como se mencionó en la sección 1.3.1 es necesario que el software de simulación pueda ejecutarse en dos plataformas Windows y Linux, por lo que al obtener la versión final del simulador en Windows fue migrado a Linux (ver Apéndice A) y se verificó que fuera funcional y cumpliera con las recomendaciones dadas a la versión en Windows.

Capítulo 6. **Resultados Obtenidos**

6.1 Introducción

Al finalizar el desarrollo de cualquier software es deseable obtener los resultados que fueron establecidos en el inicio del proyecto. En el caso del SSPC, realizar un análisis de los resultados obtenidos es de vital importancia debido a que define el éxito del proyecto.

6.2 Resultados

El primer objetivo del SSPC fue aplicar una metodología de desarrollo de software que garantice la calidad del mismo, debido a los problemas presentados en la sección 2.2, no fue posible aplicar una metodología rígidamente, por lo que este objetivo se cumplió sin aplicar dicha metodología de manera estricta.

Por lo tanto, el SSPC cumple con tres objetivos:

1. La simulación con tres tipos de herramienta, en la cual los resultados se muestran en tres ventanas.
2. Obtención de los tiempos de estancia que la herramienta sólida pule sobre cada zona de la superficie de vidrio, visualizando los resultados en cuatro ventanas.
3. Diseño de la herramienta de pétalo para obtener el desgaste deseado, mostrando los resultados en tres ventanas iguales a las del punto 1.

A continuación, se examinan los resultados para cada uno de estos objetivos.

6.2.1 Simulación del Proceso de Pulido

El software de simulación realiza el proceso de pulido con tres herramientas: sólida, anular y de pétalo. En el SSPC, el usuario puede elegir en el menú (ver Figura 6.1) o en la barra de herramientas (Figura 6.2), la herramienta de trabajo con la que realizará la simulación.

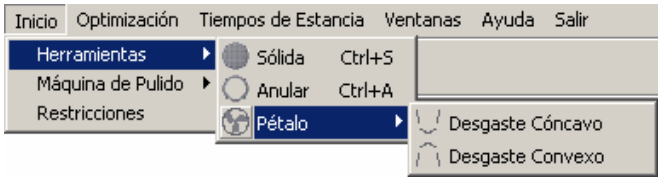


Figura 6.1 Menú de herramientas.

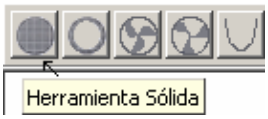


Figura 6.2 Barra de herramientas.

Posteriormente, aparecerá una pantalla en la cual deberá ingresar los datos de los parámetros de pulido tal y como se muestra en la Figura 6.3, la que muestra la ventana de parámetros de una herramienta anular. Finalmente, al indicarle al SSPC que se desea iniciar la simulación emergerán tres gráficas (ver Figura 6.4) representando 1) el desgaste generado por la herramienta, ventana 1, visto desde un corte transversal, 2) la oscilación del borde de la herramienta sobre el vidrio, ventana 2, en la cual el técnico podrá darse una idea del tamaño y la zona que abarcará la herramienta respecto al vidrio y 3) la forma de la herramienta (ventana 3), elegida por el usuario, de acuerdo al valor de los parámetros de pulido.

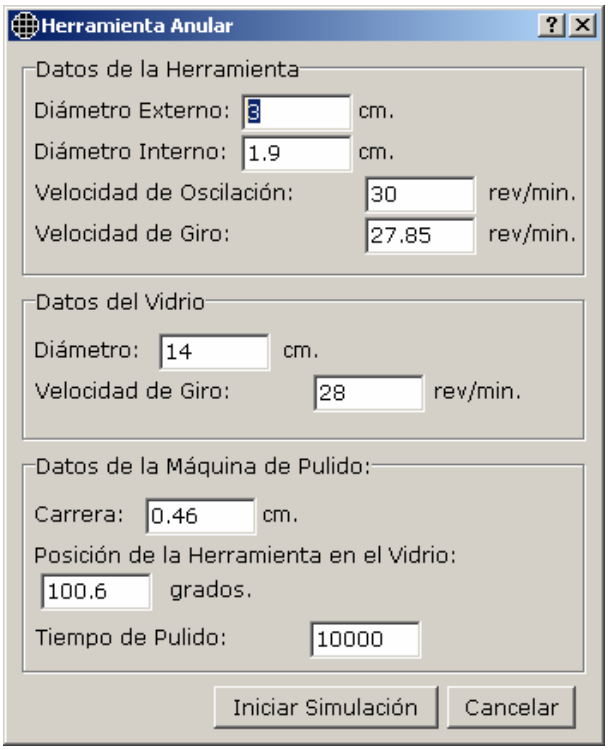


Figura 6.3 Ventana de parámetros de pulido para una herramienta anular.

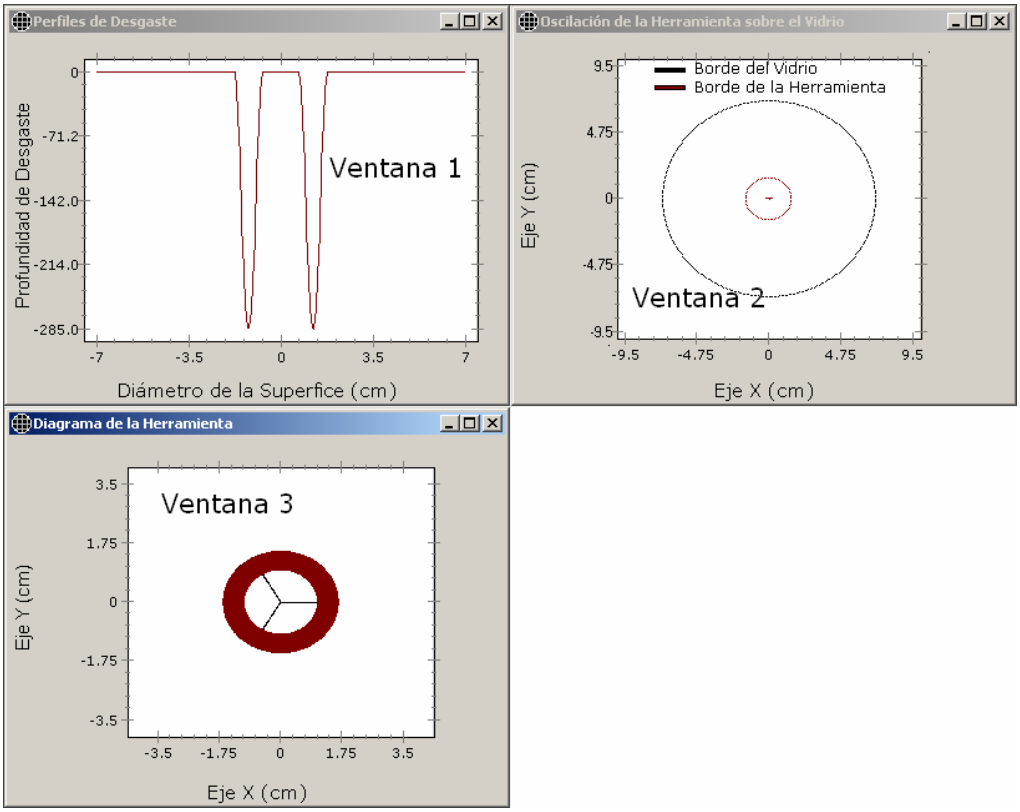


Figura 6.4 Gráficas de desgaste.

Cabe mencionar, que el usuario podrá visualizar en todo momento los valores de los parámetros a la derecha de su pantalla, Figura 6.5. Algunos de los resultados obtenidos se muestran en las siguientes secciones.

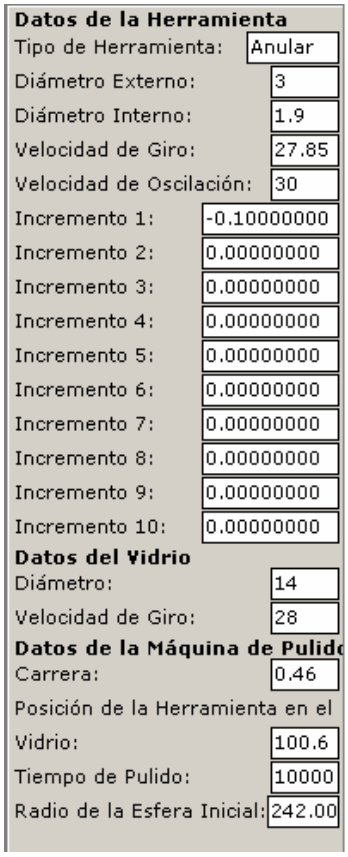


Figura 6.5 Parámetros de pulido.

Los parámetros utilizados en las siguientes simulaciones son valores que pueden ser usados en un taller de pulido, es decir, valores que típicamente los técnicos e investigadores emplean.

6.2.1.1 Herramienta Sólida

Se simuló el pulido de una superficie con una herramienta sólida con los siguientes valores de parámetros:

- Datos de la herramienta

- Diámetro: 9 cm.
- Velocidad angular de giro 29 rpm[3].
- Velocidad angular de oscilación: 30 rpm

- Datos del vidrio
  - Diámetro: 15 cm.
  - Velocidad angular de giro de 29 rpm.

El SSPC obtiene tres gráficas tal y como se muestra en la Figura 6.6 en la cual se visualiza en la ventana 1, la gráfica de desgaste generado en el vidrio (visto desde un corte transversal hecho a la superficie de vidrio) por la herramienta de la ventana 3; finalmente, en la ventana 2 se muestra la simulación de la oscilación del borde de la herramienta sobre el vidrio.

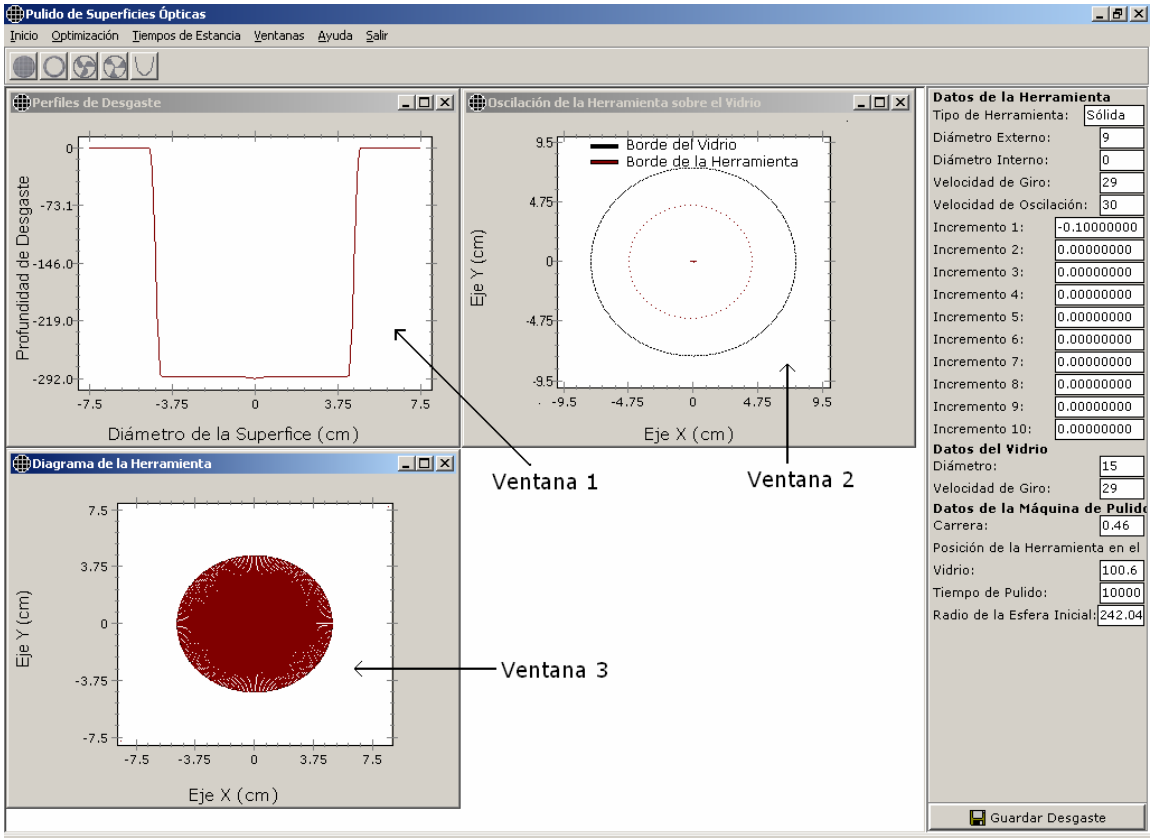


Figura 6.6 Simulación de Pulido con una Herramienta Sólida (Windows).

[3] rpm_ revoluciones por minuto.

En caso del simulador de la plataforma Linux (con los mismos datos), se obtuvieron los mismos resultados definidos por cada una de las gráficas tal y como se observa en la Figura 6.7.

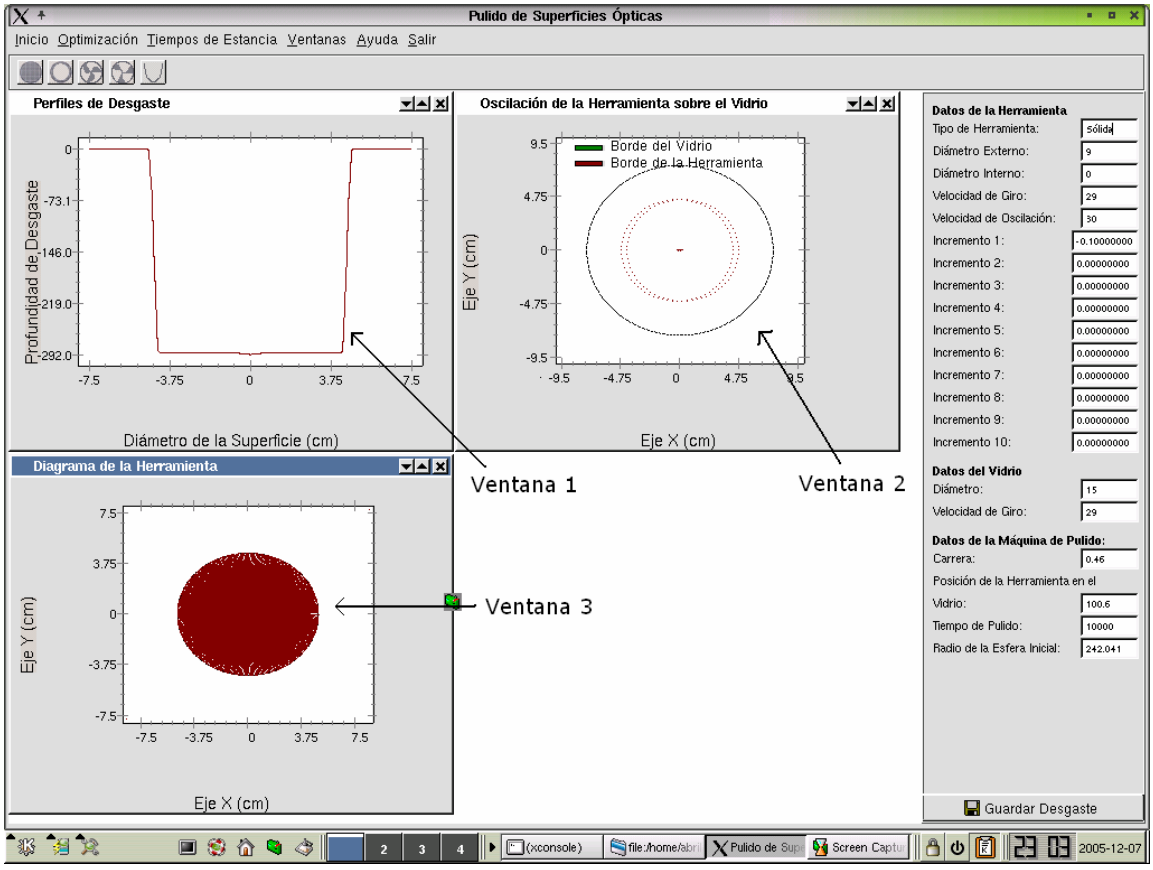


Figura 6.7 Simulación de Pulido con una Herramienta Sólida (Linux).

A continuación, se describe la simulación del proceso de pulido con una herramienta anular.

6.2.1.2 Herramienta Anular

Al simular el pulido clásico haciendo uso de una herramienta anular con los siguientes datos:

- Datos de la herramienta
  - Diámetro externo: 5 cm.
  - Diámetro interno: 2 cm.
  - Velocidad angular de giro: 32 rpm.

- Velocidad angular de oscilación 50 rpm.

- Datos del vidrio
  - Diámetro: 10 cm.
  - Velocidad angular de giro: 32 rpm.

Se obtienen los resultados mostrados en la Figura 6.8, al igual que con la herramienta sólida en las que el usuario podrá conocer los resultados que obtendrá al utilizar una herramienta anular con las características presentadas con anterioridad.

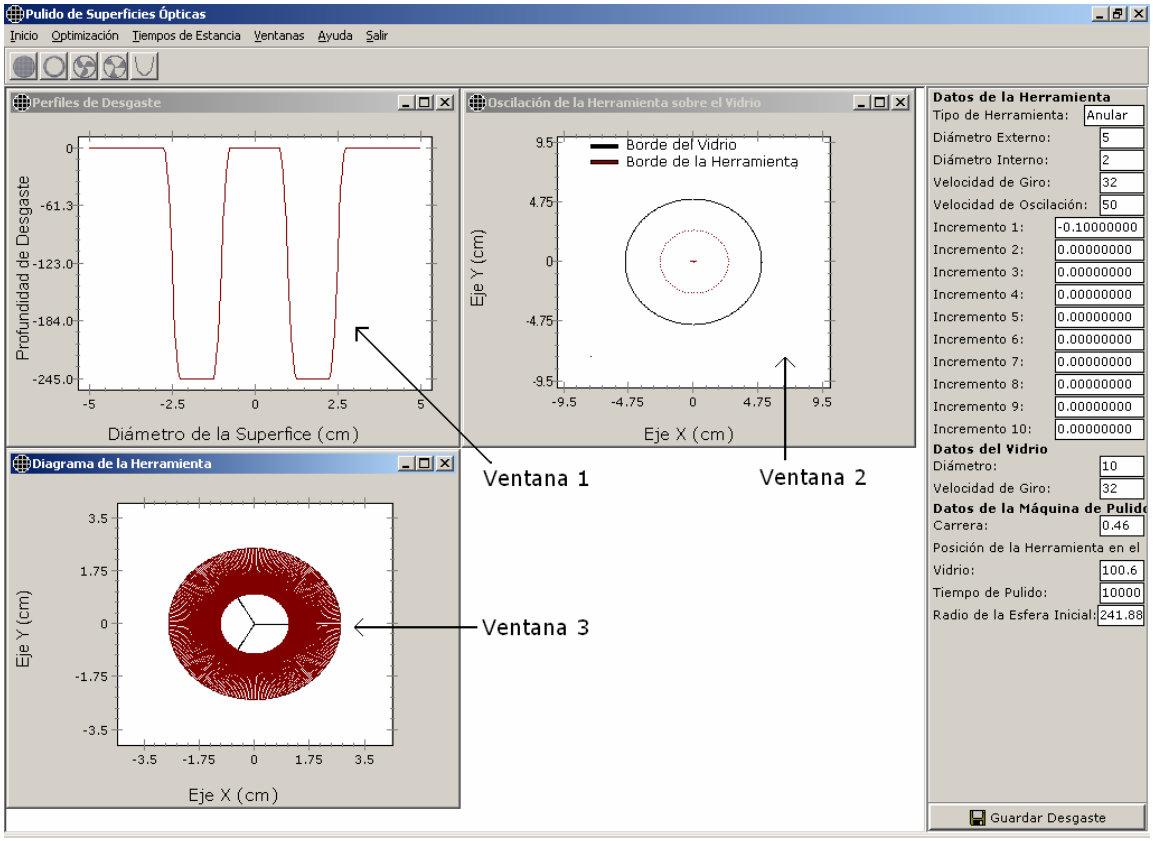


Figura 6.8 Simulación de Pulido con una Herramienta Anular (Windows).

Igualmente, en Linux se obtienen los mismos resultados mostrados en la figura anterior, como se visualiza en la Figura 6.9.

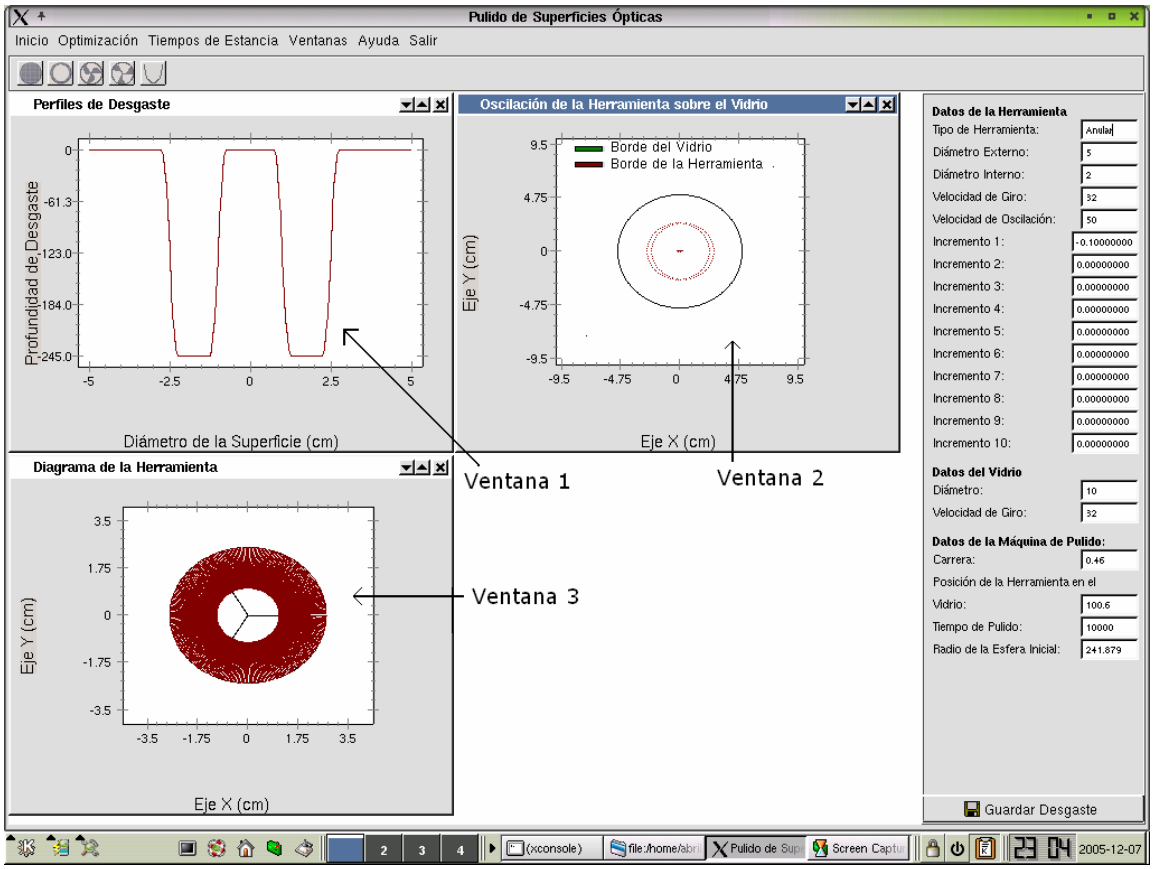


Figura 6.9 Simulación de Pulido con una Herramienta Anular (Linux).

En la siguiente subsección, se presenta las simulaciones con la herramienta de pétalo.

6.2.1.3 Herramienta de Pétalo – Superficie Cóncava

En el software de simulación, se puede simular el desgaste de dos tipos de superficies (cóncava y convexa) con la herramienta de pétalo. Al simular una herramienta de pétalo para fabricar una superficie cóncava con las siguientes características:

- Datos de la herramienta
  - Diámetro: 13 cm.
  - Velocidad angular de giro: 27.85 rpm.
  - Velocidad angular de oscilación: 30 rpm.

- Incrementos: 1 (anillo ubicado en el centro) al 10 (incremento ubicado a la orilla) de -0.09, -0.27, -0.36, -0.45, -0.81, -0.99, -1.17, -1.35, -1.53, -1.71.

- Datos del vidrio
  - Diámetro: 14 cm.
  - Velocidad angular de giro: 28 rpm.

Se obtienen del SSPC las gráficas que se presentan en la Figura 6.10.

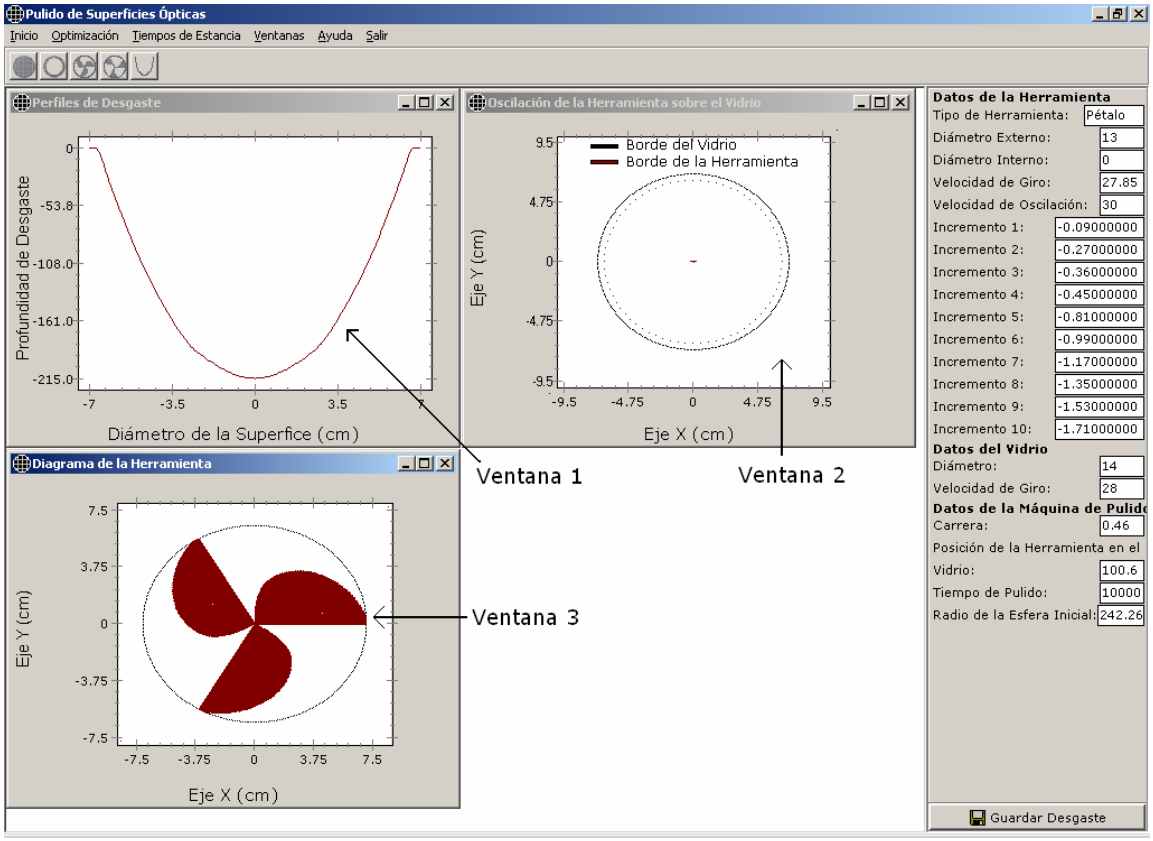


Figura 6.10 Simulación de Pulido con una Herramienta de Pétalo – Superficie Cóncava (Windows).

En la Figura 6.11 se muestran los resultados obtenidos en la plataforma Linux, como se ve en la figura las gráficas son idénticas.

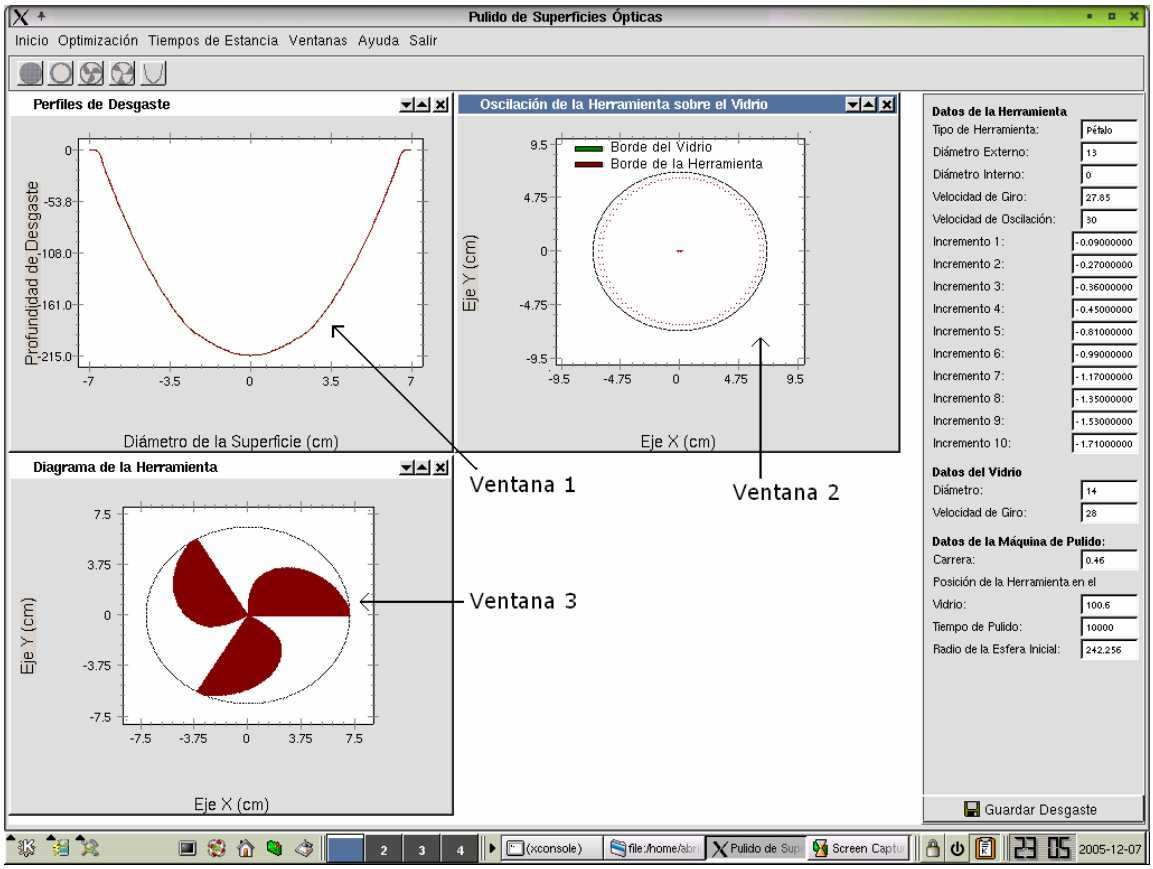


Figura 6.11 Simulación de Pulido con una Herramienta de Pétalo – Superficie Cóncava (Linux).

6.2.1.4 Herramienta de Pétalo – Superficie Convexa

Para generar una superficie convexa, se seleccionó a una herramienta de pétalo de con valores de parámetros como se presentan a continuación:

- Datos de la herramienta
  - Diámetro: 16 cm.
  - Velocidad angular de giro: 30 rpm.
  - Velocidad angular de oscilación: 40 rpm.
  - Incrementos: 1 (anillo ubicado en el centro) al 10 (incremento ubicado a la orilla) de 0.08, 0.2, 0.5, 0.6, 0.7, 0.8, 0.9, -0.2, -0.5, -0.9.
- Datos del vidrio
  - Diámetro: 17 cm.
  - Velocidad angular de giro: 31 rpm.

Se obtienen los resultados mostrados en la Figura 6.12.

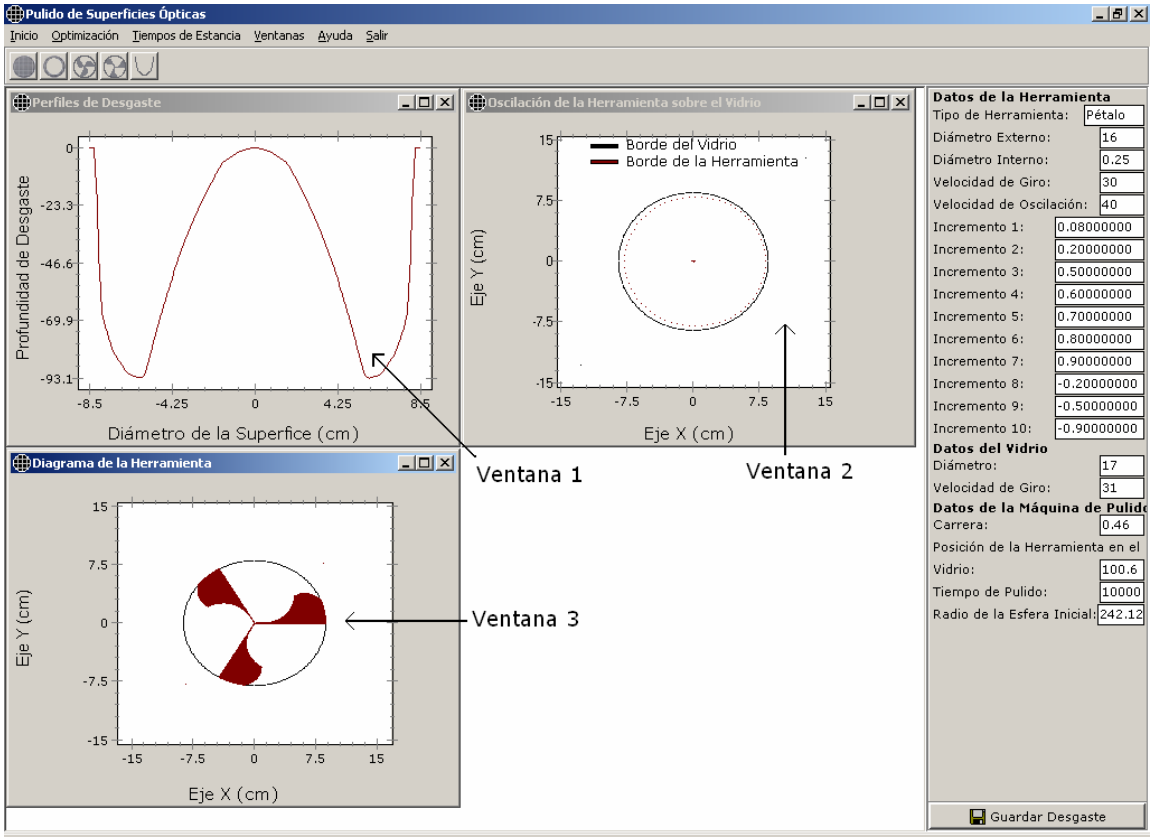


Figura 6.12 Simulación de Pulido con una Herramienta de Pétalo – Superficie Convexa (Windows).

De la misma manera, en Linux se obtienen resultados idénticos a los de la plataforma Windows tal y como se muestra en la Figura 6.13.

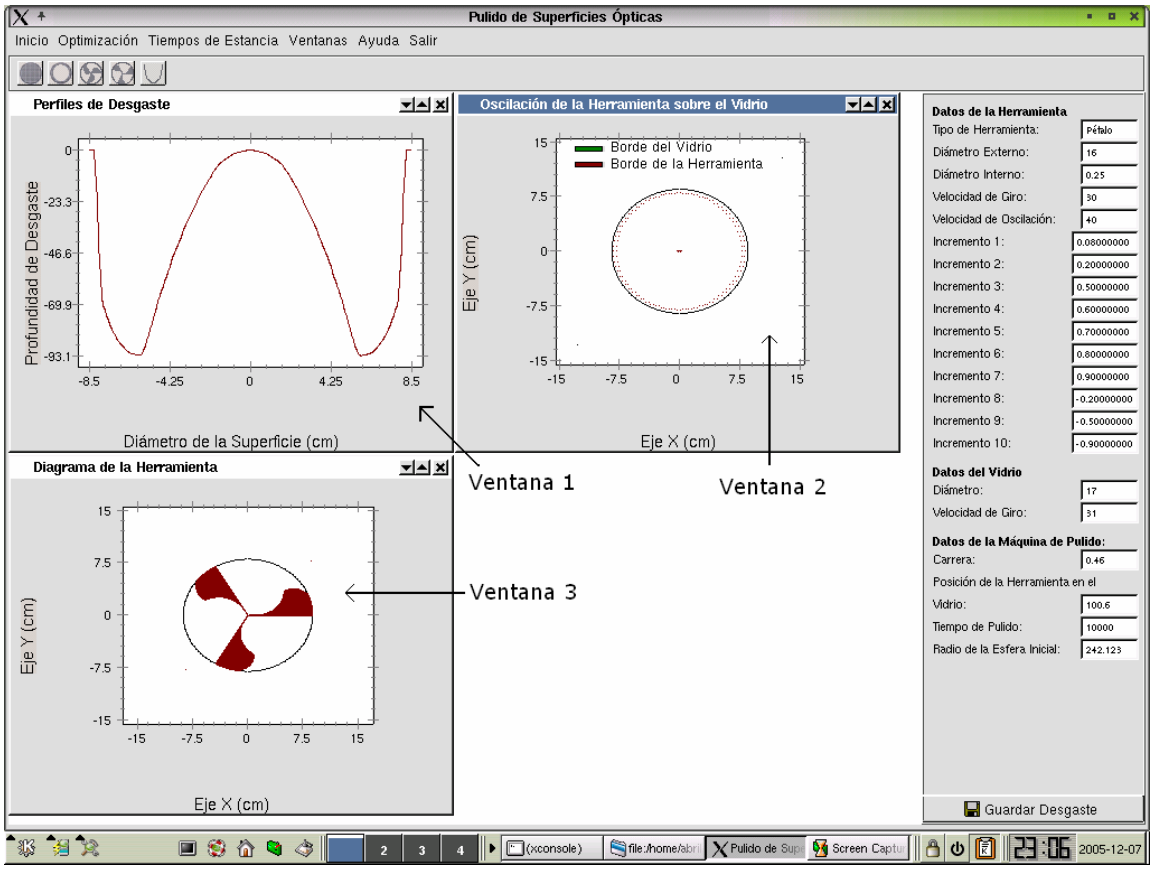


Figura 6.13 Simulación de Pulido con una Herramienta de Pétalo – Superficie Convexa (Linux).

En la siguiente sección, se describen los resultados del SSPC en la obtención de los tiempos de estancia de la herramienta sólida sobre el vidrio.

6.2.2 Tiempos de Estancia

Como se mencionó en el capítulo 4, se obtuvieron los tiempos que una herramienta sólida permanecerá trabajando sobre cada zona de la superficie de vidrio. El usuario elige en el menú de herramientas la opción “Tiempos de Estancia” y la superficie que desea generar (cóncava o convexa), ver Figura 6.14.

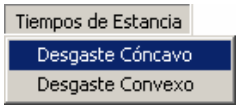


Figura 6.14 Opción de tiempos de estancia.

En seguida, el SSPC solicita los parámetros de la cónica a generar, Figura 6.15, el usuario indica el momento en que inicia la simulación y obtiene como resultado cuatro gráficas que representan:

1. La esfera de referencia generada por el SSPC en comparación con el desgaste que se desea generar de acuerdo a los parámetros introducidos por el usuario.
2. La diferencia que existe entre el desgaste deseado y la esfera de referencia (cuanto más se aleje del cero significa que se debe pulir aún más en esa zona de la superficie).
3. Los tiempos de estancia obtenidos para cada posición del vidrio.
4. El desgaste generado aplicando los tiempos en el pulido de la superficie.

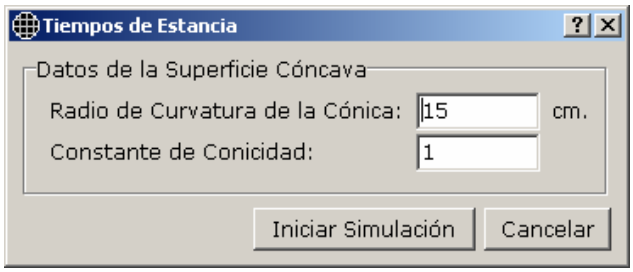


Figura 6.15 Ventana de parámetros para una superficie cóncava.

A continuación, se detallan los resultados obtenidos al generar los dos tipos de superficies.

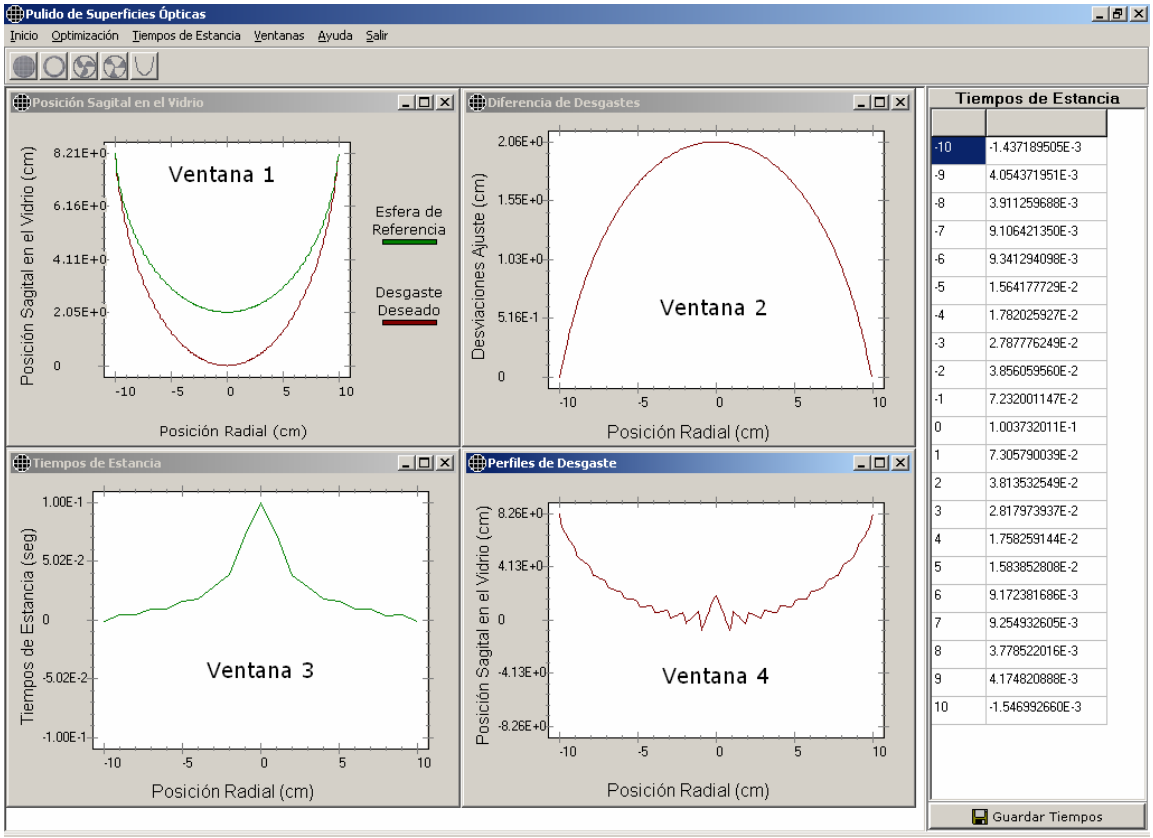
6.2.2.1 Superficie Cóncava

Supongamos que deseamos generar una superficie cóncava elipsoidal, con los siguientes valores:

- Radio de curvatura paraxial: $R = 14.28cm.$
- Constante de conicidad de $K = 1$.

El SSPC obtiene como resultados cuatro ventanas, ver Figura 6.16, en la ventana 1 se puede notar que existe mayor diferencia de ajuste en el centro de la superficie que en los bordes; la ventana 2, ilustra el desajuste entre ambas

superficies; la ventana 3 muestra los tiempos de estancia de la herramienta sólida en diferentes puntos de la superficie, como se puede ver, el tiempo es máximo en el centro que es donde se requiere más desgaste, y la ventana 4 representa el perfil obtenido al aplicar mínimos cuadrados, el cual, como se puede observar se aproxima a la gráfica de la ventana 1. Dado que en algunas simulaciones se obtenían tiempos menores, ya no se realizó un análisis de error en el ajuste. En trabajos futuros, se seleccionará otra técnica de optimización.



-10	-1.437189505E-3
-9	4.054371951E-3
-8	3.911259688E-3
-7	9.106421350E-3
-6	9.341294098E-3
-5	1.564177729E-2
-4	1.782025927E-2
-3	2.787776249E-2
-2	3.856059560E-2
-1	7.232001147E-2
0	1.003732011E-1
1	7.305790039E-2
2	3.813532549E-2
3	2.817973937E-2
4	1.758259144E-2
5	1.583852808E-2
6	9.172381686E-3
7	9.254932605E-3
8	3.778522016E-3
9	4.174820888E-3
10	-1.546992660E-3

Figura 6.16 Resultados del Simulador - Tiempos de Estancia Desgaste Cóncavo (Windows).

Para asegurar que el software de simulación en la plataforma Linux obtuviera los mismos resultados que en Windows se ejecutó el mismo ejemplo y se encontraron los mismos tiempos, al igual que las gráficas mencionadas con anterioridad, ver Figura 6.17.

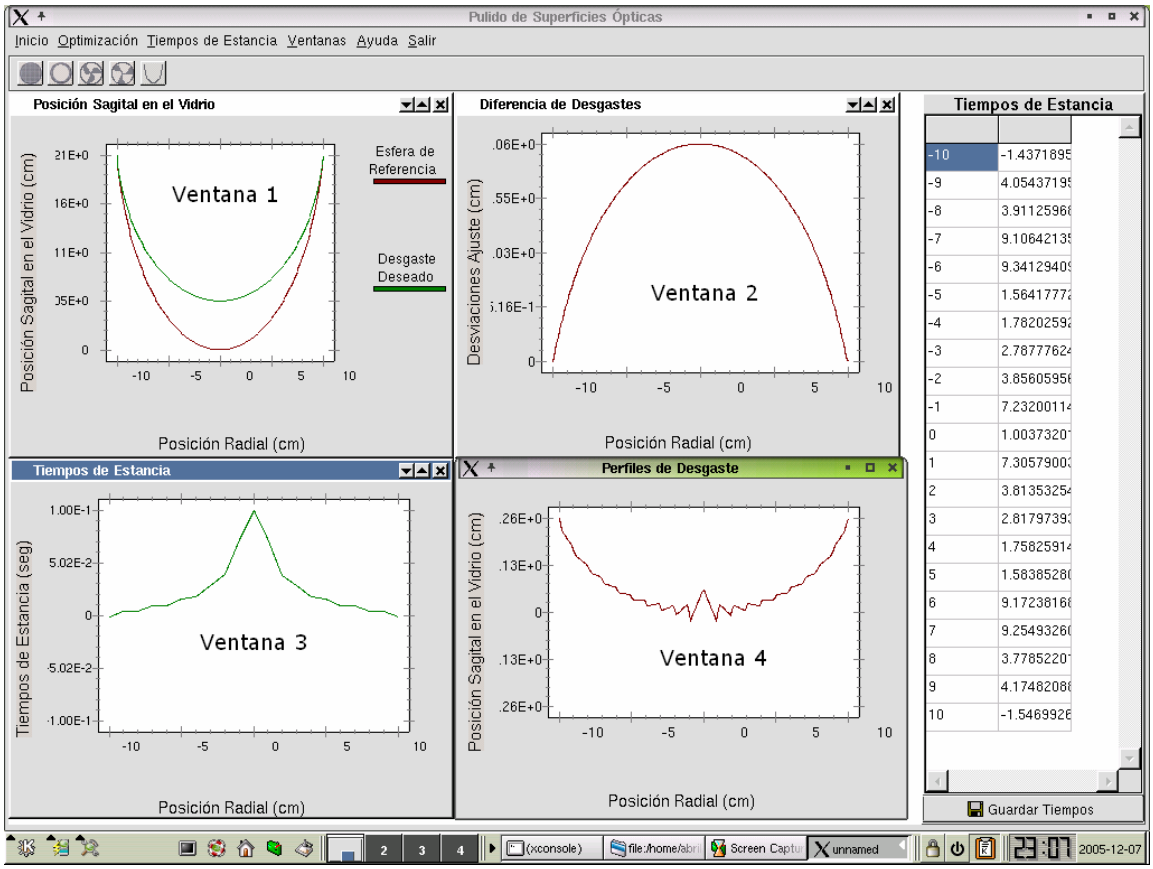


Figura 6.17 Resultados del Simulador - Tiempos de Estancia Desgaste Cóncavo (Linux).

6.2.2.2 Superficie Convexa

Si se simula el pulido de una superficie parabólica convexa con las siguientes características:

- Radio de curvatura: $R = 16.66\,cm.$
- Constante de conicidad $K = -5$

Se obtienen los resultados que se presentan en la Figura 6.18, en la cual se pueden observar las gráficas mencionadas en la sección 6.2.2, con la diferencia que se trata de una superficie convexa, en lugar de pulir más en el centro de la superficie, se debe pulir aún más en los bordes.

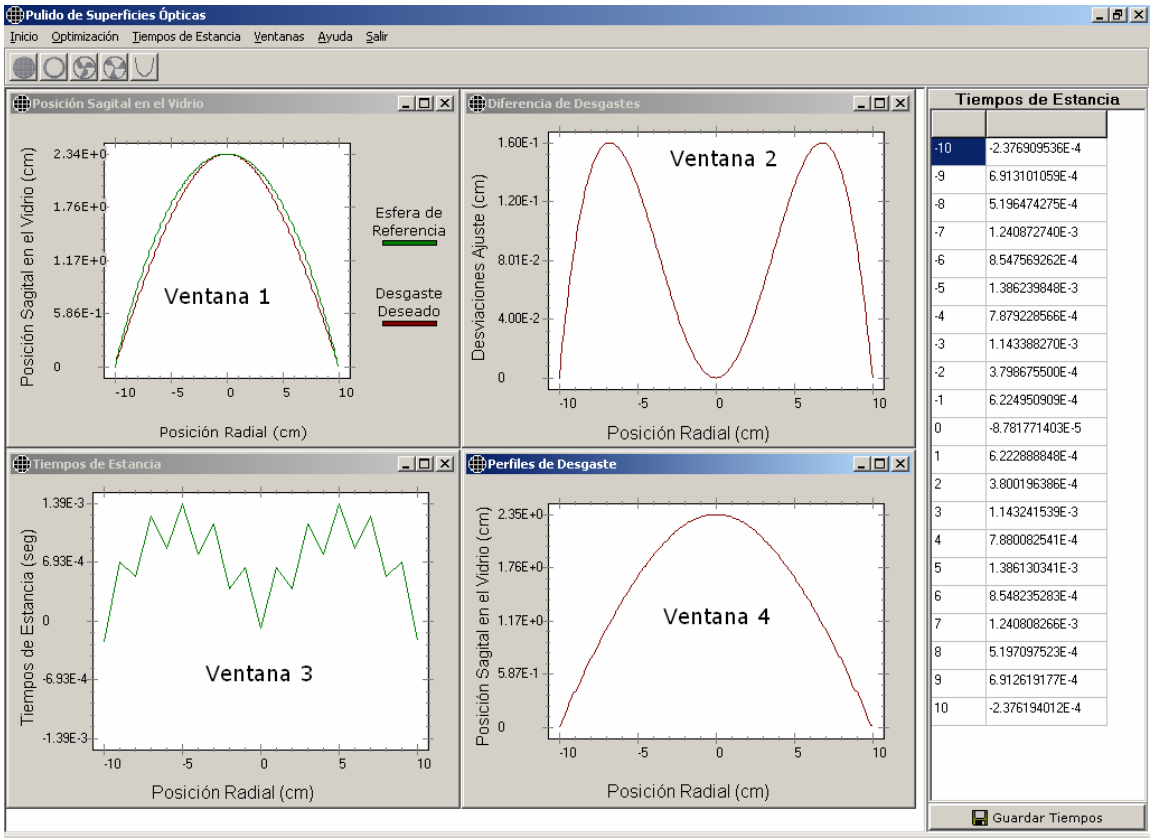


Figura 6.18 Resultados del Simulador - Tiempos de Estancia Desgaste Convexo (Windows).

Al igual que en Windows, se ejecutó el simulador en Linux con los mismos datos introducidos para comprobar de que se comporta de la misma manera y se obtuvo el resultado que muestra la Figura 6.19.

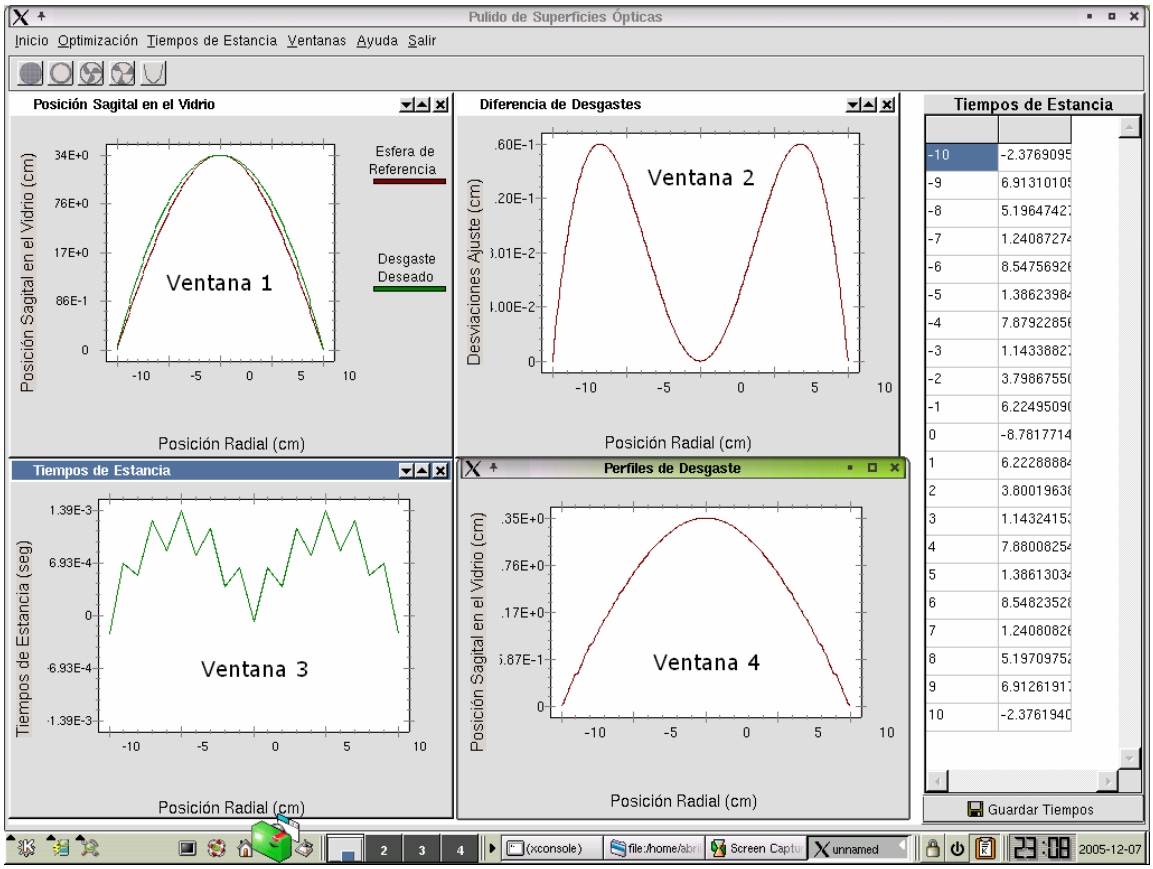


Figura 6.19 Resultados del Simulador - Tiempos de Estancia Desgaste Convexo (Linux).

Finalmente, se detallan los resultados obtenidos en el diseño de la forma de la herramienta de pétalo para generar la superficie deseada.

6.2.3 Perfil de la Herramienta de Pétalo con AG

El último de los objetivos específicos presentados en la sección 1.7.1 es obtener la forma de la herramienta de pétalo. El usuario elige la opción ”Optimización”, en seguida elige “Herramienta de Pétalo – Algoritmos Genéticos” tal y como se muestra en la Figura 6.20, el usuario ingresa los datos y al realizar la simulación emergerán tres gráficas similares a las de simulación con las tres herramientas de pulido.

Figura 6.20 Menú para optimización de la herramienta de pétalo.

En las dos plataformas (Windows y Linux) se calcula el valor RMS mencionado en el capítulo 4; cuando este valor tiende a cero, significa que el desgaste obtenido se asimila al deseado, el técnico decidirá si le conviene utilizar dicha herramienta o realiza otra simulación con el objetivo de obtener un RMS menor.

Por otro lado, es importante mencionar que los parámetros utilizados en el algoritmo genético son lo que se presentan a continuación, los cuales no podrán ser modificados por los usuarios:

1. Longitud del cromosoma, la cual, quedó determinada por el rango en el cual se calculan las m_j y la precisión requerida, tal y como lo menciona Zbigniew [26]. Utilizando un rango de [-5, 5] y una precisión de dos cifras después del punto decimal, por lo que la longitud del cromosoma requerida fue de 10 bits.
2. La probabilidad de cruza fue de 1, este valor se obtuvo debido a que F. J. Cuevas et al. [27] menciona que un valor típico de la probabilidad de cruza es un valor cercano a uno y se optó por tomar este valor.
3. La probabilidad de mutación utilizada fue de 0.05, ya que la probabilidad de mutación es inversamente proporcional al tamaño de la población, en este caso de 20 individuos, lo cual nos da una probabilidad de mutación de 0.05 [20], este valor cae dentro del rango típico de valores de probabilidad de mutación [27] de [0.001, 0.1] y cumple con ser un valor pequeño [28].
4. El número máximo de generaciones que se eligió fue de 10, debido a que al ejecutarse 10 veces en una sola corrida el AG (sección 4.3), esto se incrementa en 100 generaciones y el tiempo en que realiza la optimización se incrementa.

Al simular el proceso de pulido con los siguientes parámetros:

- Datos de la herramienta:
  - Diámetro: 13 cm.
  - Velocidad de giro: 27.85 rpm.
  - Velocidad de oscilación: 30 rpm.
  - Amplitud de oscilación: 0.5 cm.
- Datos del vidrio

- Diámetro: 14 cm.
- Velocidad de giro: 28 rpm.

Como se puede observar, los resultados obtenidos tanto en Windows como en Linux son similares como lo demuestran los RMS obtenidos (ver Figuras 6.21 y 6.22).

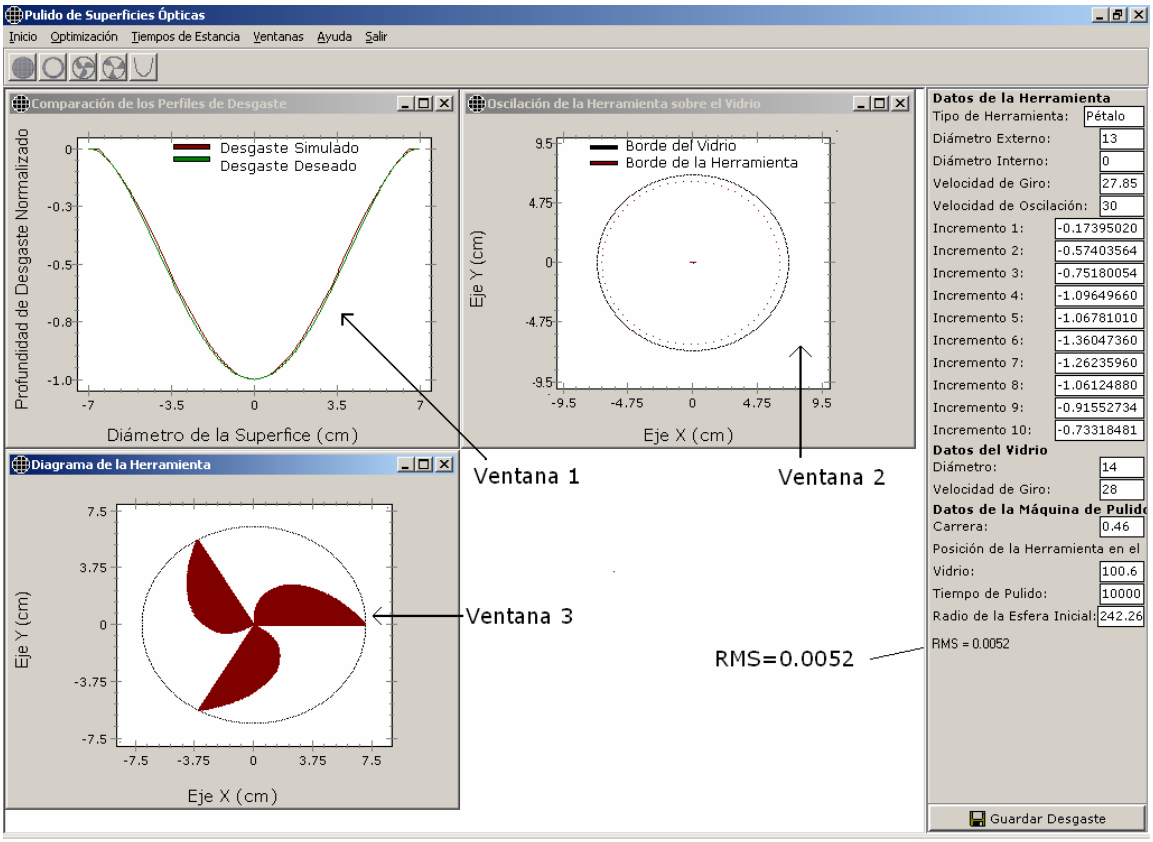


Figura 6.21 Optimización de la herramienta de pétalo (Windows).

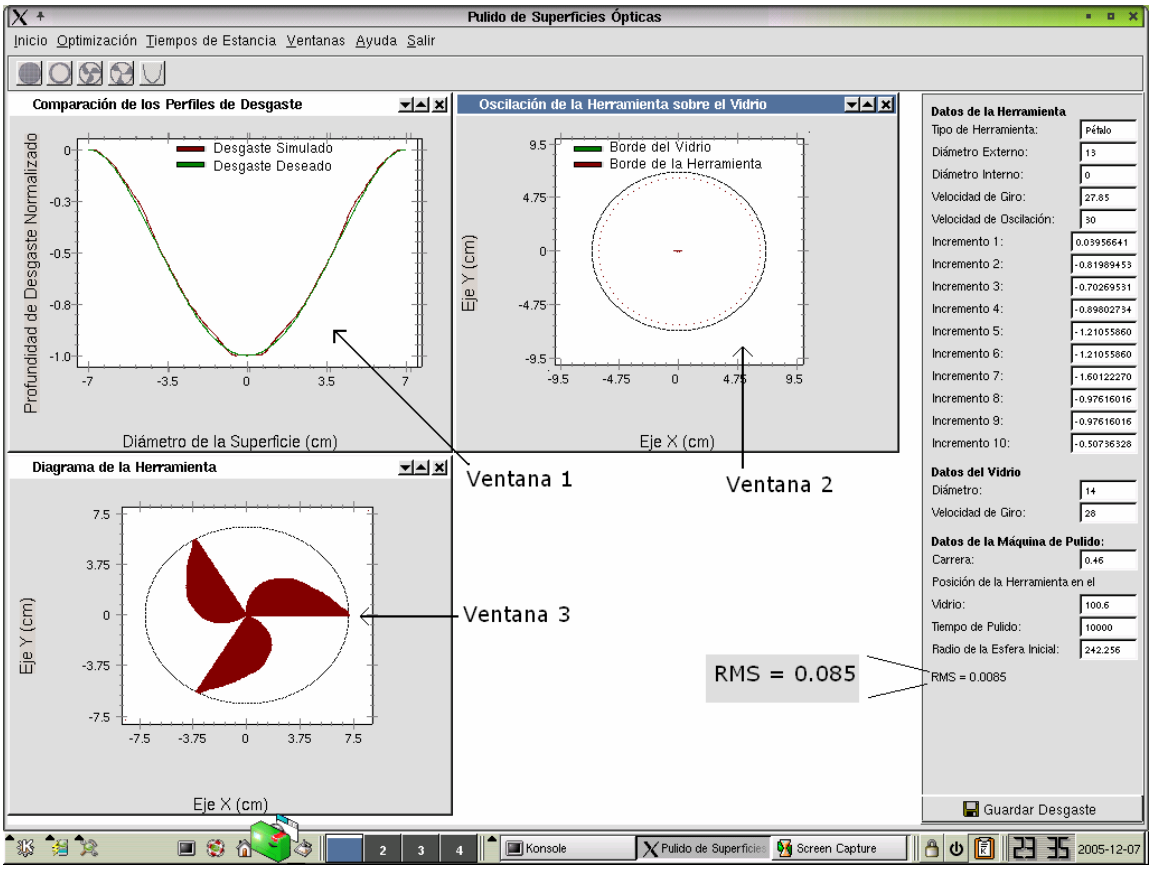


Figura 6.22 Optimización de la herramienta de pétalo (Linux).

Capítulo 7. **Conclusiones y Trabajo Futuro**

7.1 Conclusiones

El objetivo general del presente trabajo fue desarrollar un software de simulación de pulido de superficies utilizando el método clásico (SSPC), que servirá de soporte técnico a los talleres de óptica para determinar los parámetros necesarios de construcción y obtener el desgaste deseado en el generado de una superficie óptica, por lo que se puede señalar que se cumplió con este objetivo. Este SSPC puede ser ya utilizado por los técnicos de pulido, sin embargo, para enriquecerlo es necesario agregar más parámetros de pulido y complementar las investigaciones realizadas en esta área (ver sección 1.4).

Al iniciar el desarrollo del SSPC, se planteó seguir cada una de las tareas de RUP, pero debido a las condiciones y alcances del tema de tesis no se consideraron los flujos de trabajo secundarios. El flujo de trabajo realizado abarcó el modelado del negocio, requisitos, análisis y diseño, implementación, pruebas y despliegue. Las tareas no realizadas fueron configuración y gestión de cambios, gestión del proyecto y entorno. Por lo tanto, las fases aplicadas fueron concepción (inicio), elaboración, construcción y transición.

Un problema importante que surgió casi al final del desarrollo de la segunda versión del SSPC, fue que los encargados del proyecto no tenían una visión clara de lo que querían realmente de este software, por lo que los requerimientos iniciales no fueron cumplidos en su totalidad y cambiaron constantemente a lo largo de todo el desarrollo de este proyecto de tesis. Pero como lo marca RUP, es necesaria la constante comunicación del usuario para satisfacer sus necesidades, por lo que se deben realizar tres iteraciones para llegar a un documento de requisitos completo.

La difícil comunicación con el usuario y la distancia existente entre la BUAP y la UTM provocaron que la mayoría de los requerimientos fueran establecidos por los investigadores involucrados en el proyecto, y no por los usuarios a los cuales esta dirigido este simulador. Por consiguiente, el SSPC no se centró en cubrir las necesidades básicas de los técnicos al manejar términos y unidades de parámetros diferentes a los empleados por los investigadores.

Debido a que mínimos cuadrados minimiza las diferencias al cuadrado, estas pueden ser positivas o negativas repercutiendo en los tiempos positivos como negativos, lo cual no siempre es deseable (cuando alguno de los tiempos es negativo significa que en lugar de quitar material del vidrio, se necesita agregar más material a la superficie de vidrio y esto en la práctica no es posible). Por lo que el método de mínimos cuadrados no es adecuado totalmente para la solución del problema inverso en el pulido de superficies.

Algoritmos Genéticos como método de optimización nos permitió calcular el perfil de la herramienta de pétalo que genera el desgaste más cercano al deseado.

Al realizar el análisis de las pruebas realizadas al SSPC, los usuarios hicieron notar que en el proceso de pulido se toman en cuenta aún más parámetros de los que se tenían contemplados, tales como el tipo de vidrio que se está utilizando, tolerancias de pulido, peso, tiempo real de trabajo, tipo de abrasivo, entre otros y que es necesario incluirlos. Para incluir estos parámetros, es necesario realizar pruebas experimentales y obtener una constante que deberá incluirse en el SSPC para generar simulaciones que predigan resultados reales, sin embargo, esto no se contempla en el presente proyecto de tesis. No obstante, aún con estos problemas, este software puede ser utilizado por los técnicos de pulido.

A pesar de los problemas, descritos a lo largo de esta tesis se cumplió con los objetivos específicos que son:

1. Aplicar un proceso de desarrollo de software que en este caso fue RUP. El software fue modelado con UML e implementado con una tecnología orientada a objetos.
2. La integración del proceso de pulido clásico con tres tipos de herramientas: sólida, anular y de pétalo, a este proyecto de tesis. Calcular los tiempos de estancia que

una herramienta sólida debe estar sobre cada zona de la superficie de vidrio y así generar el desgaste deseado utilizando mínimos cuadrados.

3. Por último, diseñar la forma de la herramienta de pétalo utilizando como método de optimización algoritmos genéticos, para obtener el desgaste deseado sobre una superficie.

Finalmente, es importante resaltar que los resultados obtenidos por el SSPC son predicciones de desgastes cualitativos y no cuantitativos. El técnico de pulido podrá hacer uso del SSPC para saber solamente que tipo de desgaste obtendrá al utilizar una determinada herramienta y la forma de la herramienta de pétalo para generar un desgaste específico.

Cabe mencionar que este trabajo de tesis ha sido aceptado y publicado en el Congreso Nacional de Física celebrado del 17 al 21 de Octubre de 2005 en la ciudad de Guadalajara, Jalisco y en el 3er. Congreso Nacional de Ciencias de la Computación llevado a cabo del 23 al 25 de Noviembre de 2005 en la ciudad de Puebla, Puebla (ver Apéndice G).

Parte del trabajo de mejoramiento al SSPC, se describe en el siguiente apartado.

7.2 Trabajo Futuro

Cambiar el método de mínimos cuadrados por algoritmos genéticos o programación lineal para corregir el problema de tiempos de estancia de la herramienta sólida sobre la superficie del vidrio, pues algunas de las soluciones encontradas con este método contienen valores negativos, que para casos reales no pueden considerarse como solución ya que indica que en lugar de pulir la superficie se le agregue más material haciéndolo imposible en la práctica.

Al concluir el proyecto de tesis, se espera que sirva como una de las bases necesarias para que en un futuro se automatice el proceso de pulido de superficies, en el cual los técnicos no tendrán que hacer todo el trabajo de pulido, lo que implica reducir tiempos y costos en el generado de superficies ópticas. La computadora estará conectada a la

máquina pulidora y el proceso se realizará automáticamente, siendo ésto el objetivo final a alcanzar del proyecto “Pulido Predecible”.

Referencias

[1] González, García Jorge, *"Pulido Predecible de Superficies Ópticas"*, Universidad Tecnológica de la Mixteca, http://www.utm.mx/~rruiz/seminarios/docs/s8Jl.pdf, consulta: 10 de septiembre de 2005.

[2] Malacara, Daniel, *"Optical Shop Testing"*, Instituto Nacional de Astrofísica, Óptica y Electrónica. Pág. 47, (1978).

[3] A. Parra – Flores, A.Cordero-Dávila, J.Cuautle-Cortés, C.Robledo-Sánchez, J.González-García, and. V.Cabrera-Peláez, *"Simulación de desgastes en el pulido de superficies con la ecuación de Preston"*, in 46th Congreso Nacional de Física de la Sociedad Mexicana de Física Sup. Bol. Soc. Mex. Fis. 49 #5, 138 (2003).

[4] F. W. Preston, *"The Theory and Design of Plate Glass Polishing Machines"*, J. Soc. Glass Technol. 11, 214-256 (1927).

[5] Pressman, Roger. *"Ingeniería de Software. Un enfoque práctico"*, McGraw-Hill. Cuarta Edición, Pág. 17, (1998).

[6] Sommerville, Ian, *"Software Engineering"*, Addison-Wesley,.Quinta Edición, Pág. 24-41, (2002).

[7] Amescua, Antonio de; García, Luis; Martínez, Paloma; Díaz, Paloma, *"Ingeniería del Software de Gestión – Análisis y Diseño de Aplicaciones"*, Paraninfo, Pág. 29–39, (1995).

[8] Kruchten, Philippe, *"The Rational Unified Process an Introduction Second Edition"*, Addison-Wesley, Segunda Edición, Pág. 66-79, (2000).

[9] Quatrani, Terry, *"Visual Modeling With Rational Rose 2000 and UML"*, Addison-Wesley, Pág. 19-141, (2000).

[10] Departamento de Auditoria Informática, UNAM, *"Metodologías de Ingeniería de Software"* Subdirección de Sistemas Dirección General de Servicios de cómputo Académico. http://sistemas.dgsca.unam.mx/publica/pdf/metodologias.PDF , consulta: noviembre de 2005.

[11] Alcalá, Alfonso y Alfonseca, Manuel, *"Programación Orientada a Objetos – Teoría y técnicas OOP para desarrollo de software"*, Anaya Multimedia, Primera Edición, Pág. 14-19, (1992).

12] Kendall y Kendall, *"Análisis y Diseño de Sistemas"*, Pearson Educación, Tercera Edición, Pág. 860, (1997).

[13] Cordero Dávila, Luna Aguilar, Núñez Alfonso, González García, Cabrera Peláez, Valdez. Hernández, Martínez García, García Ramírez, Salas L. Cruz González, Ruíz E., Sohn E. and Rodríguez P*., "Polishing TIM Mirror Segment with HyDra"*, Proc. SPIE, vol. 4840, pp. 604 – 661, (2003).

[14] R. E. Wagner and R. R. Shannon, *"Fabrication of Aspherics Using a Mathematical Model for Material Removal"*, Applied Optics, vol. 13, pp. 1683-1689, (1974).

[15] Burden, Richard y Faires, J. Douglas, *"Análisis Numérico"*, International Thomson Editores, Sexta Edición, Pág. 476, (1998).

[16] Chapra Steven C., Canale Raymond P., *"Métodos Numéricos para Ingenieros"*, McGrawHill, Tercera Edición, Pág. 490, (1999).

[17] Bermúdez López, Cordero Dávila, and Cuautle Cortés*, "Diseño para la construcción de Herramienta de Pétalo aplicada al pulido de superficies ópticas"*, Proc. in 46th Nac. Conf. of Physics of Physic Society, (2003).

[18] Cordero Dávila, Cabrera Peláez, Cuautle Cortés, González García, Robledo Sánchez and Bautista Elivar, *"Experimental results and wear predictions with free-pinned petal tools",* Applied. Optics 44, 1434 – 1441, (2005).

[19] Santiago A., Agustin, *"Obtención de la forma de superficies convexas asféricas con razón focal menor a 1"*, Tesis de Doctorado, Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México, Pág. 21-37, 45-50 (2001).

[20] Goldberg, David E, *"Genetic Algorithms in Search Optimization and Machine Learning"*, Reading Massachusetts, Addison Wesley Publishing Company, Pág. 2-32, (1989).

[21] Merelo, Guervós Juan Julián, *"Informática Evolutiva: Algoritmos Genéticos"*, http://geneura.ugr.es/~jmerelo/ie/ags.htm, consulta: 18 de Octubre de 2005.

[22] Santo, Orcero David, "Los algoritmos genéticos", http://www.orcero.org/irbis/disertacion/node189.html, consulta: 18 de octubre de 2005.

[23] Dumas, D. Joseph and Redish, C. Janice, *"A practical Guide to Usability Testing"*, Intellect Books, Pág. 4-5, (1999).

[24] Manchón, Eduardo, "¿Qué es Usabilidad?", http://www.gestiopolis.com/canales5/ger/ainda/2.htm, consulta: septiembre de 2005.

[25] Nielsen, Jakob , *"Usability Engineering"*, Morgan Kaufmann, Pág. 185, (1993)

[26] Zbigniew Michalewicz, *"Genetic Algorithms + Data Structures = Evolution Programs"*, Springer, (1992).

[27] F. J. Cuevas, J. H. Sossa – Azuela and M. Servin, *"A parametric method applied to phase recovery from a fringe pattern based on a genetic algorithm",* Opt. Commun., 203, 213-223 (2002).

[28] Lance Chanbers, *"Practical Handbook of Genetic Algorithms Applications"*, Volumen 1, CRC Press, (1995).

Glosario

Abstracción: Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar *cómo* se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando los están, una variedad de técnicas son requeridas para ampliar una abstracción [11].

Clases: descripción de un grupo de objetos con propiedades, comportamiento, relación con otros objetos y semántica en común [8].

Directa: toma elementos de acuerdo a un criterio objetivo, como son «los x mejores», «los x peores»... los del tipo y se selecciona el primero por un método aleatorio o estocástico [21].

Encapsulamiento: También llamada "ocultación de la información", esto asegura que los objetos no pueden cambiar el estado interno de otros objetos de maneras inesperadas; solamente los propios métodos internos del objeto pueden acceder a su estado. Cada tipo de objeto expone una *interfaz* a otros objetos que especifica cómo otros objetos pueden interactuar con él. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción [11].

Equiprobable: todos tienen la misma probabilidad de ser escogidos [21].

Escaños: se divide el rango del número aleatorio en un número predeterminado de escaños. Estos se reparten de acuerdo con la ley d'Hont, tomando como «puntuación» para repartir los escaños el grado de adaptación [21].

Facilidad de uso: facilidad con la que los nuevos usuarios pueden tener una interacción efectiva [26].

Flexibilidad: variedad de facilidades con las que el usuario y el sistema pueden intercambiar información [26].

Función base: desgaste de la superficie producido por una herramienta sólida en una zona determinada de la superficie.

Función deseo: desviaciones que existen de una superficie esférica de referencia y la superficie cónica que deseamos fabricar punto a punto.

Herencia: Organiza y facilita el polimorfismo y la encapsulación permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que reimplementar su comportamiento. Esto suele hacerse habitualmente agrupando los objetos en *clases* y las clases en *árboles* o *enrejados* que reflejan un comportamiento común [11].

Interfaz: Comunicación del software con el usuario que debe ser amigable y entendible para él.

Polimorfismo: Las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del referente. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama *asignación tardía* o *asignación dinámica*. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++ [11].

Programación Lógica: se define porque las instrucciones que lo componen no tienen un orden de ejecución determinado, sino que reciben el control de un motor de inferencia [10].

Programación Orientada a Objetos: la ejecución de un programa se desencadena por medio de un mensaje que alguien como el usuario, un programa u objeto envía a un objeto determinado [10].

Programación Procedimental: caracterizada esencialmente porque las instrucciones que lo componen se ejecutan secuencialmente en un orden preestablecido [10].

Requerimientos: describen una condición o capacidad sobre la cual un sistema debe conformarse; son derivados directamente de los usuarios, desde un contrato, estándares, o de especificaciones formales de un documento.

Restos Estocásticos: similar al de escaños, sólo que los escaños no asignados directamente se asignan de forma aleatoria [21].

Robustez: nivel de apoyo al usuario que facilita el cumplimiento de los objetivos del software. [26]

Rueda de la Ruleta: definimos un rango similar a la de sorteo. El número al azar será un aleatorio menor que el rango. El elemento escogido, será aquel en cuyo rango esté el número resultante de sumar el número aleatorio con el resultado total que sirvió para escoger el elemento anterior [21].

Sorteo: cada individuo de la población tiene asignado un rango proporcional a su adaptación. Se escoge un aleatorio dentro del rango global, y el escogido es el que tenga dicho número dentro de su rango [21].

Torneo: escoge un subconjunto de individuos de acuerdo con una de las técnicas anteriores y de entre ellos selecciona el más adecuado por otra técnica [21].

Apéndice A

Migración de Windows a Linux

Al haber finalizado con el desarrollo del software cumpliendo con todos los requisitos propuestos, así como las pruebas correspondientes, se requiere la migración de la implementación de Windows a Linux. Para ello se cuenta con los visuales Borland C++Builder (Windows) y Borland Kylix (Linux). La ventaja que ofrecen estos visuales radica en que al desarrollar un sistema en una de las plataformas mencionadas proporciona la oportunidad de migrar los datos a la otra plataforma, es decir, no es necesario volver a implementar el sistema, con solo unos ajustes en el código se puede correr el sistema en otra plataforma.

A continuación se describen los pasos que se siguieron para la migración del simulador de pulido:

1. *Crear un nuevo proyecto.* Debido a que los proyectos no son compatibles de Linux a Windows, es necesario crear un nuevo proyecto.
2. *Agregar formas.* Adherir cada una de las formas utilizadas en Windows, por medio de la opción “Add to Project...” que se encuentra en la barra de herramientas de Project.
3. Cambiar la biblioteca fuente en Windows se utiliza *vcl.h* en Linux se hace uso de *clx.h* estas opciones se encuentran en los archivos con extensión cpp.
4. *Cambiar el origen de los componentes*. Esto define la fuente de las formas, botónes, etc., en Windows se utilizan “#include <Controls.hpp>” y en Linux “#include <QControls.hpp>”, al compilar el proyecto sobresalen estos errores.

5. *Cambiar de WideString a AnsiString.* En los visuales se tienen variables tipo cadena, Windows utiliza variables llamadas AnsiString que son de 8 bits, Linux a su vez hace uso de variables llamadas WideString de 16 bits, regularmente se tienen problemas en Linux porque los editores obtienen los datos en WideString y al querer convertirlos a números utilizanso las funciones StrToFloat (cadena a flotantes) o StrToInt (cadena a enteros) entre otros, pero los parámetros de estas funciones son de tipo AnsiString, por lo cual se deben convertir de la siguiente manera:

```
AnsiString Variable1;
WideString Variable2;
Variable2 = Edit1->Text;
Variable1 = AnsiString(Variable2);
```

Con los cambios mencionados anteriormente un programa sencillo desarrollado en Windows, puede trabajar correctamente en la plataforma Linux, el simulador de pulido siguió teniendo problemas al intentar hacerlo trabajar. A continuación se describen los problemas generados y la solución a dichos problemas.

Incompatibilidad de funciones

Como se sabe, funciones como *__fastcall TPrincipal::FormActivate(TObject *Sender)* o *__fastcall TPrincipal::TPrincipal(TComponent* Owner):TForm(Owner)*, son funciones generadas por el visual, las cuales son utilizadas para inicializar valores entre otras cosas, dichas funciones que de una u otra manera fueron utilizadas en C++Builder no funcionan de la misma manera en Kylix, un ejemplo de esto es que el *FormActivate* en Windows se ejecuta cada vez que la forma es activada, en cambio en Kylix ésta se ejecuta como un ciclo, por lo cual al ser utilizada como función de inicialización de variables, dichas variables no podían ser modificadas por el usuario.

Otra de las funciones que generó problemas en la migración del simulador fue *__fastcall TForm1l::TForm1(TComponent* Owner):TForm(Owner)*, debido a que ésta función solo se ejecuta una vez al presentar una forma, también fue utilizada principalmente para inicialización de valores, al ser trasladado a Linux, las variables que contenían valores de tipo flotante recibían valores enteros, por lo tanto los cálculos

realizados por el simulador siempre resultaban erróneos, en consecuencia los resultados presentados al usuario no eran correctos. Éste fue el problema principal que no dejaba avanzar en la migración de los datos.

Ejecución en diferentes distribuciones de Linux

Al correr el programa se genera un ejecutable en Linux, pero éste se ejecuta solamente cuando se compila en Kylix, usualmente al instalar Kylix se ubica en /usr/local/kylix3/bin y /usr/local/kylix3/lib, se deben agregar estas rutas en el archivo llamado ldconf ubicado en el directorio /etc, finalmente en modo de superusuario se ejecuta el comando ld.so. Si se siguen correctamente estos pasos se podrá ejecutar el programa en cualquier otra distribución de linux.

Apéndice B

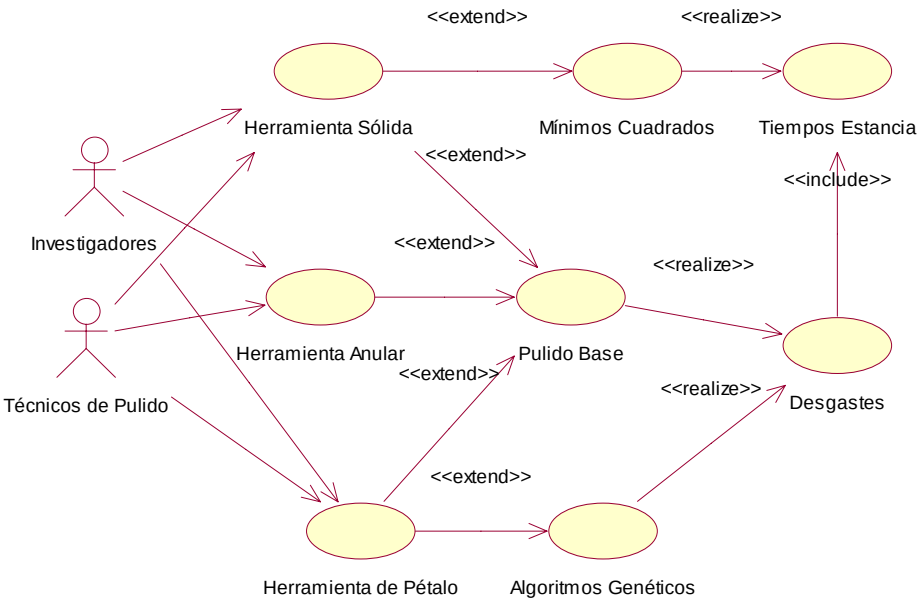
Diagrama Unificado de Casos de Uso

Diagrama Unificado de Casos de Uso

Breve Descripción

El presente documento se muestra los casos de uso localizados para el desarrollo del simulador de pulido.

Diagrama



Apéndice C

Especificación de Caso de Uso: Enseñanza de Pulido

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El caso de uso inicia cuando el Técnico no sabe como funciona el simulador de pulido por lo cual necesita de una ayuda.
2. El Técnico ingresa al menú de Inicio.
3. El Técnico elige la opción “Ayuda del Simulador”

El caso de uso finaliza cuando el técnico obtiene la ayuda necesaria del simulador.

Especificación de Realización de Caso de Uso: Herramienta Sólida

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

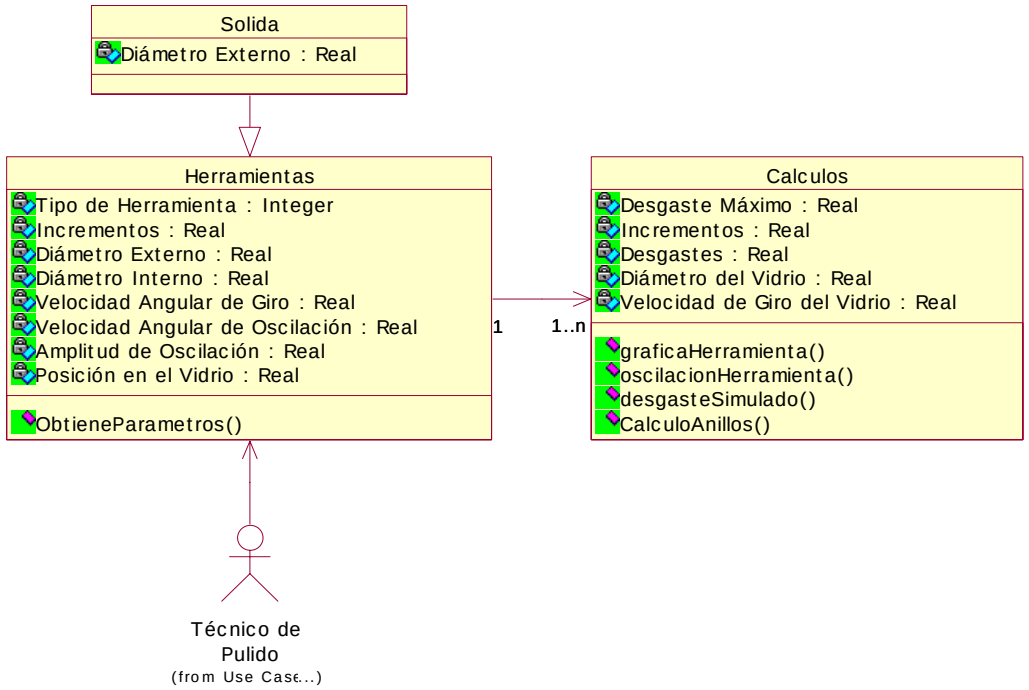
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

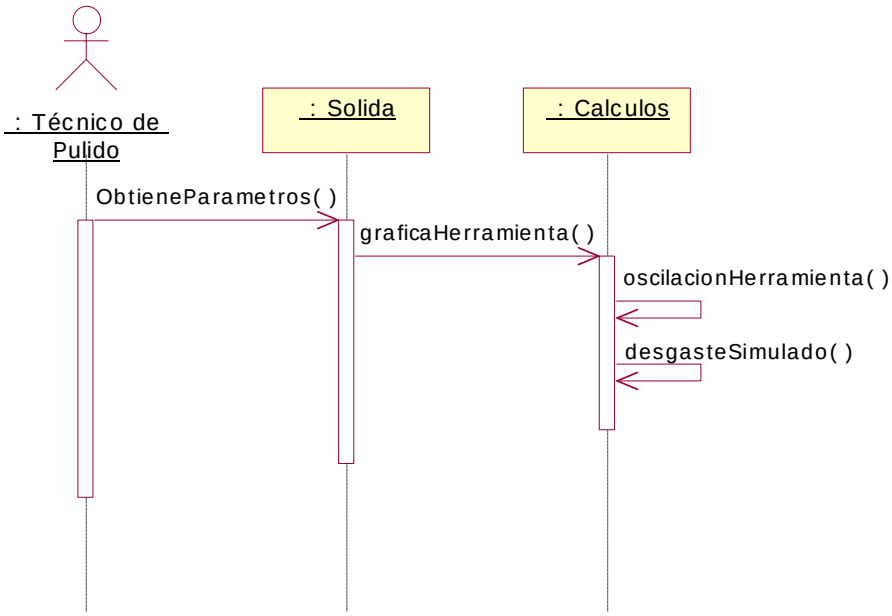
Flujo Básico

1. El caso de uso inicia cuando el Técnico requiere hacer uso de una herramienta sólida para generar una superficie.
2. El Técnico ingresa al menú de Inicio.
3. El Técnico elige la herramienta sólida
4. El Técnico ingresa los parámetros correspondientes a la herramienta: diámetro de la herramienta, diámetro del vidrio, posición de la herramienta en el vidrio, velocidades angulares y amplitud de oscilación.

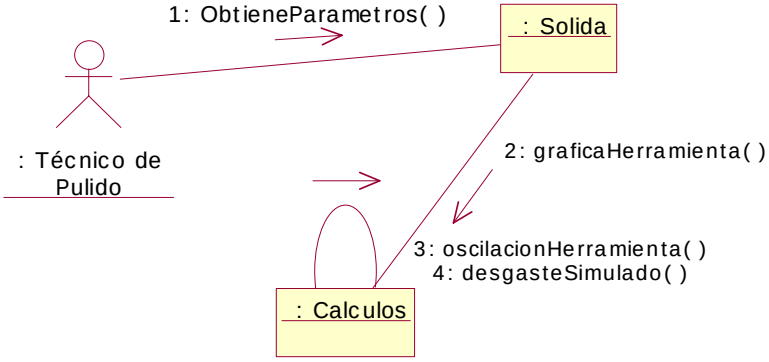
Diagrama de clases



Diagramas de secuencia



1.1 Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Herramienta Anular

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

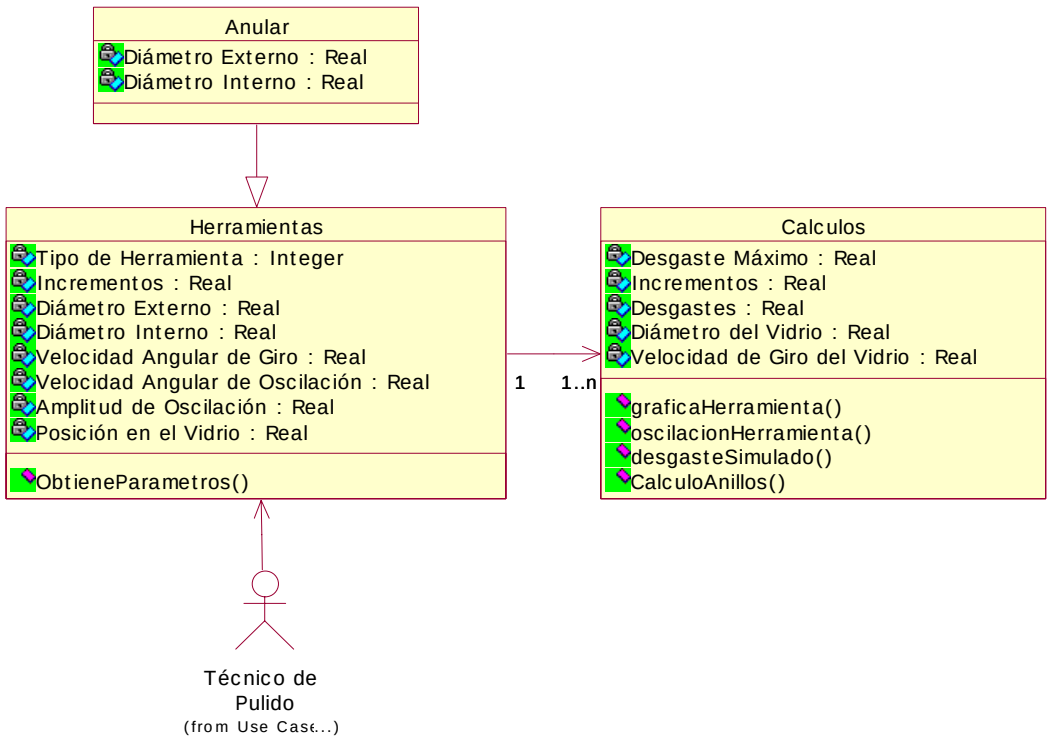
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El caso de uso inicia cuando el Técnico requiere hacer uso de una herramienta anular para generar una superficie.
2. El Técnico ingresa al menú de Inicio.
3. El Técnico elige la herramienta anular.
4. El Técnico ingresa los parámetros correspondientes a la herramienta: diámetro externo y diámetro interno de la herramienta, diámetro del vidrio, posición de la herramienta en el vidrio, velocidades angulares y amplitud de oscilación.

Diagrama de clases



Diagramas de secuencia

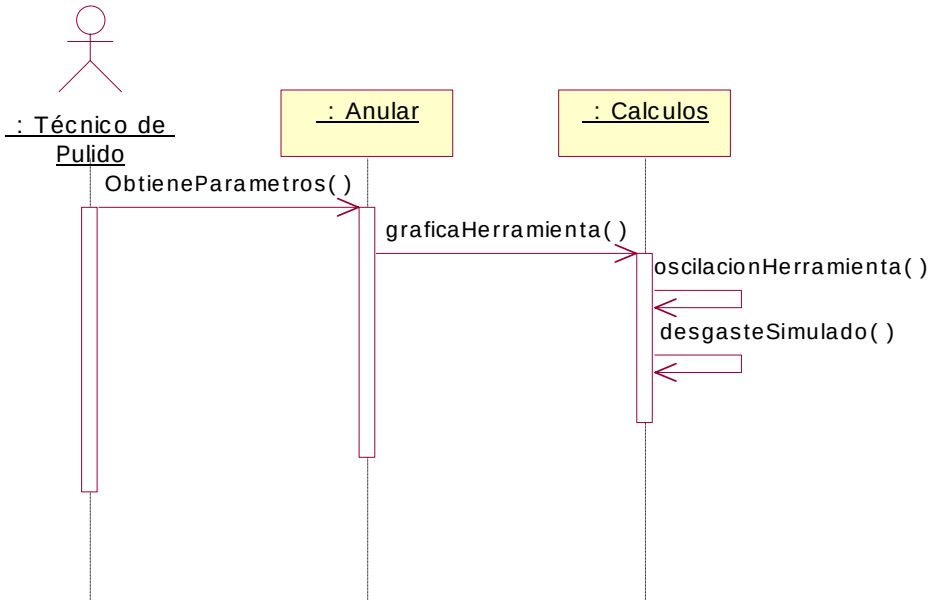
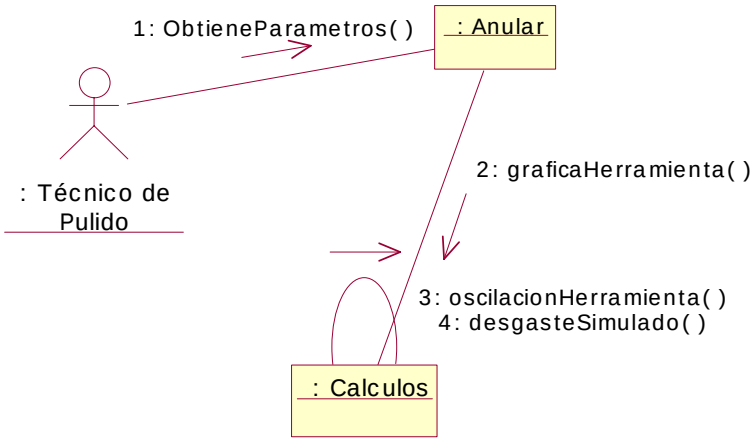


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Herramienta de Pétalo

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El caso de uso inicia cuando el Técnico requiere hacer uso de una herramienta anular para generar una superficie.
2. El Técnico ingresa al menú de Inicio.
3. El Técnico elige la herramienta de pétalo.
4. El Técnico ingresa los parámetros correspondientes a la herramienta: diámetro de la herramienta, diámetro del vidrio, posición de la herramienta en el vidrio, incrementos, velocidades angulares y amplitud de oscilación.

Flujos Alternativos

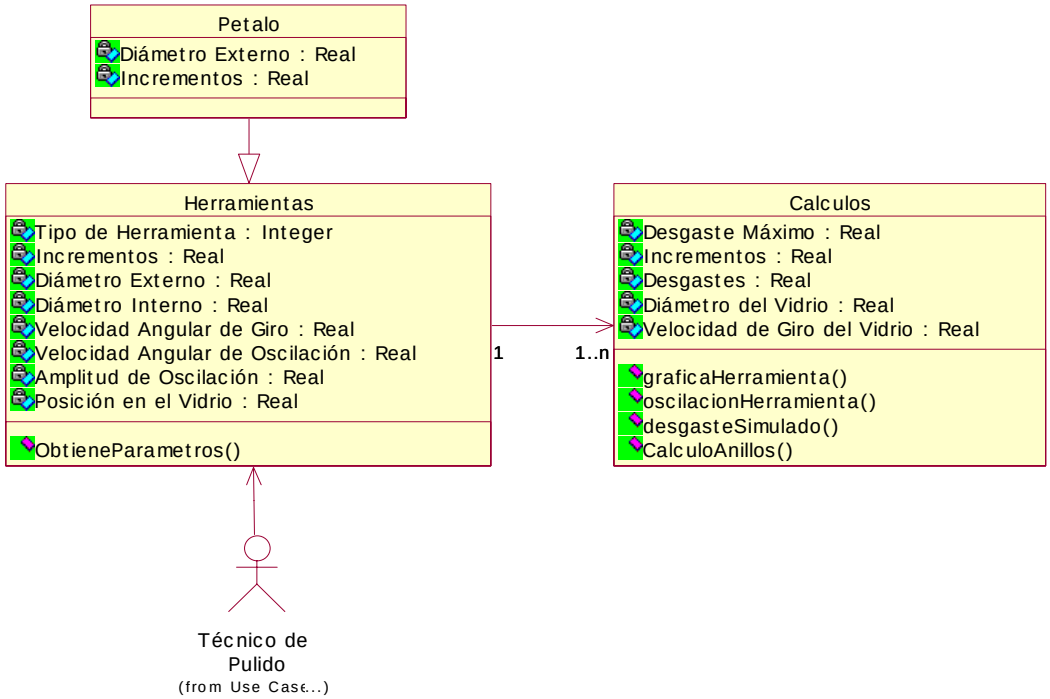
Herramienta de Pétalo para una superficie cóncava

El técnico puede realizar una superficie cóncava por lo que elige a una herramienta de pétalo para dicha superficie.

Herramienta de Pétalo para una superficie convexa

El técnico puede realizar una superficie convexa por lo que elige a una herramienta de pétalo para dicha superficie.

Diagrama de clases



Diagramas de secuencia

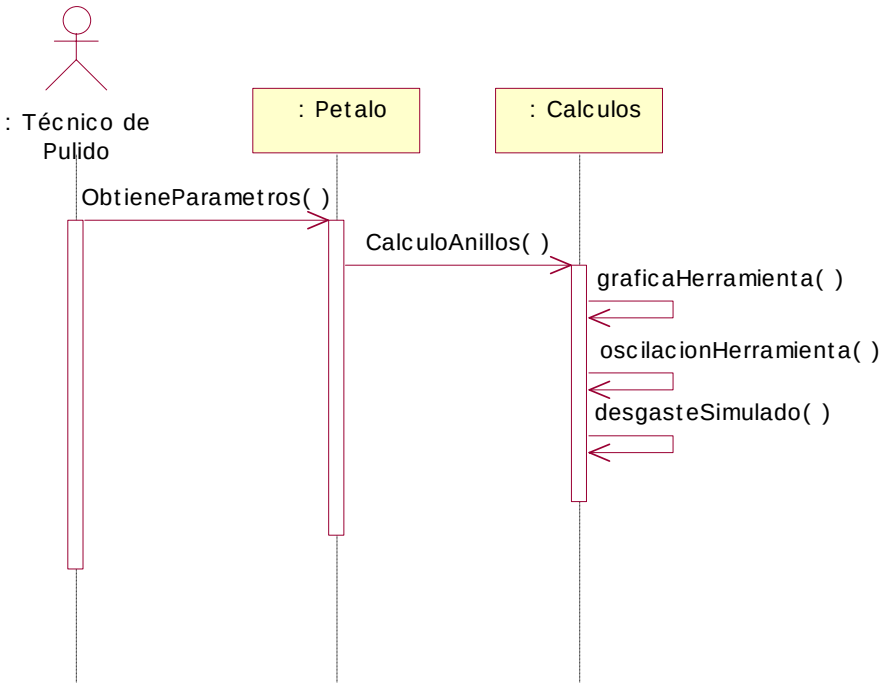
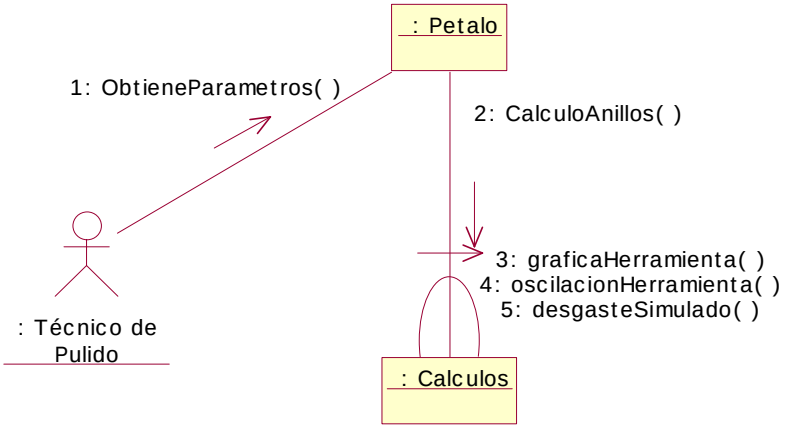


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Mínimos Cuadrados

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El técnico de pulido desea conocer los tiempos de estancia de una herramienta sólida para generar una superficie.
2. El técnico selecciona en el menú principal la opción Tiempos de estancia
3. El simulador le solicita los parámetros correspondientes a la superficie como radio de curvatura y constante de conicidad.
4. El simulador hace los cálculos correspondientes y obtiene los tiempos de estancia.

Flujos Alternativos

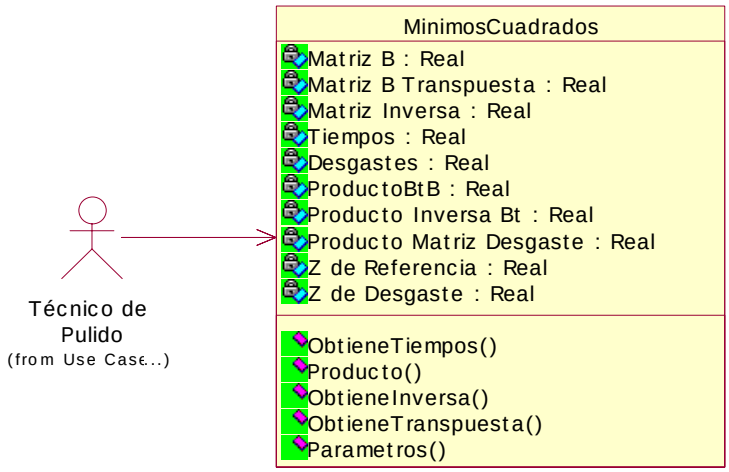
Superficie Cóncava

El técnico desea generar una superficie cóncava por lo que elige tiempos de estancia para una superficie cóncava.

Superficie Convexa

El técnico desea generar una superficie convexa por lo que elige tiempos de estancia para una superficie convexa.

Diagrama de clases



Diagramas de secuencia

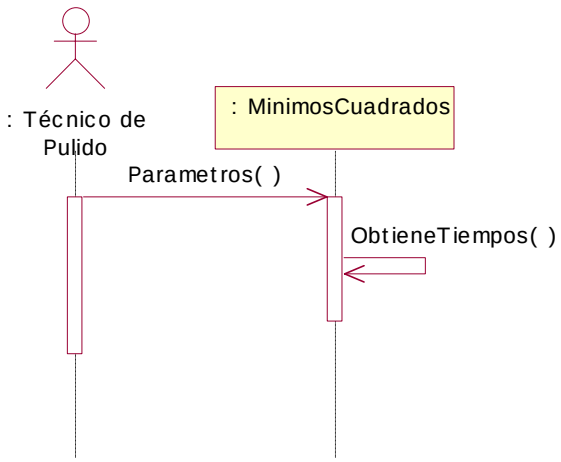
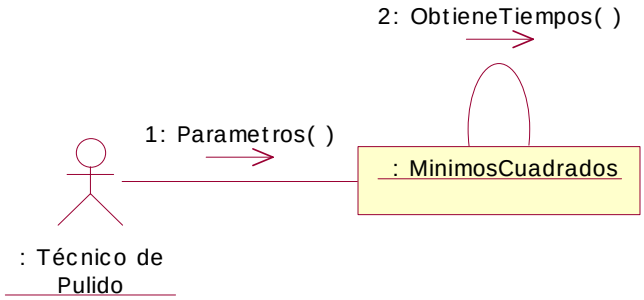


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Programa de Pulido Base

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

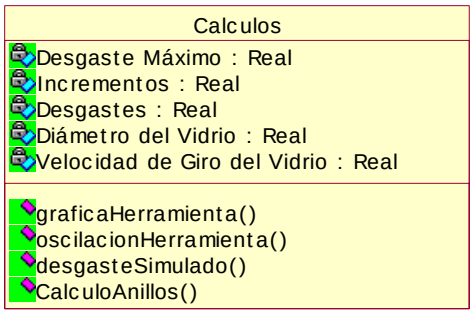
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El programa realiza los cálculos básicos necesarios en el pulido de acuerdo a la herramienta elegida por el usuario para obtener los resultados de la simulación.

Diagrama de clases



Diagramas de secuencia

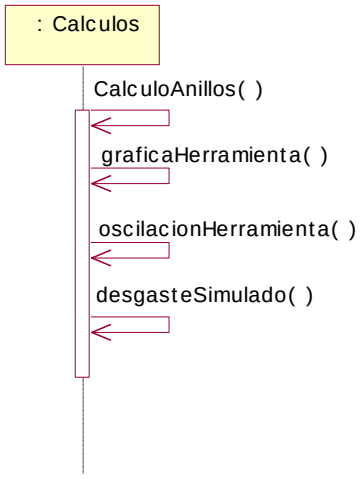
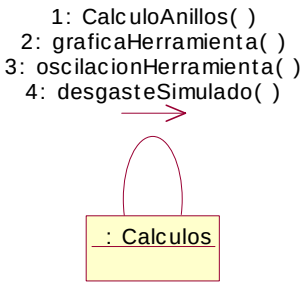


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Algoritmos Genéticos

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

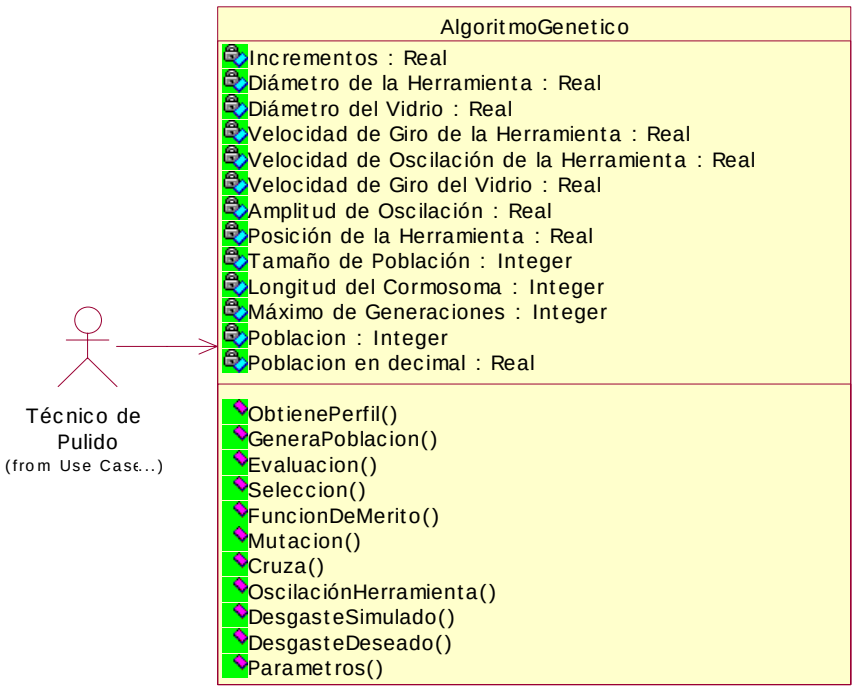
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El caso de uso comienza cuando el Técnico de Pulido quiere conocer la forma de una herramienta de pétalo para generar una superficie de vidrio específica.
2. El Técnico elige la opción Optimización de la Herramienta de Pétalo con Algoritmos Genéticos en el menú principal del simulador de pulido.
3. El simulador solicitará los parámetros necesarios para iniciar la optimización de la herramienta de pétalo.
4. El técnico ingresa los parámetros correspondientes.
5. El simulador inicia la optimización y obtiene la forma de la herramienta.

Diagrama de clases



Diagramas de secuencia

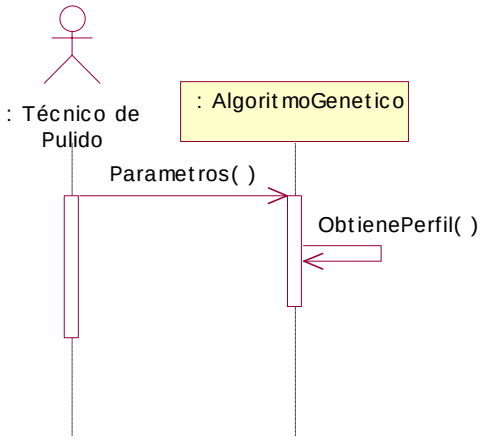
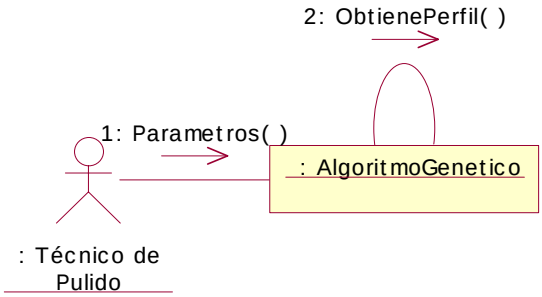


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Gráfica de Tiempos de Estancia

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

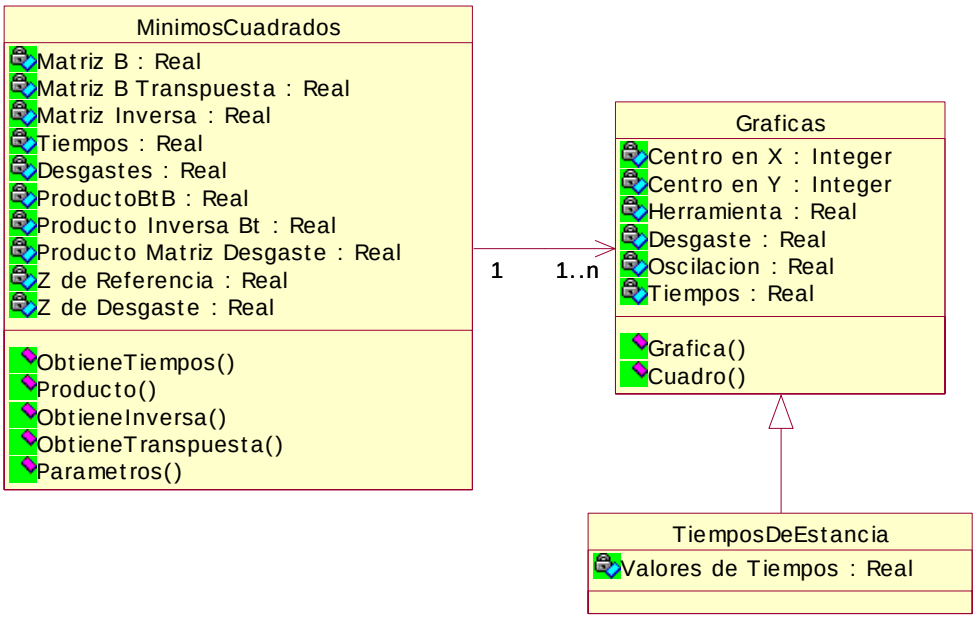
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El programa muestra por medio de una gráfica y por un archivo de texto, los tiempos en los cuales debe la herramienta actuar sobre cada intervalo de la superficie de vidrio, el desgaste obtenido con los tiempos obtenidos, la diferencia de desgaste de una esfera de referencia con la superficie deseada y posición sagital en el vidrio.

Diagrama de clases



Diagramas de secuencia

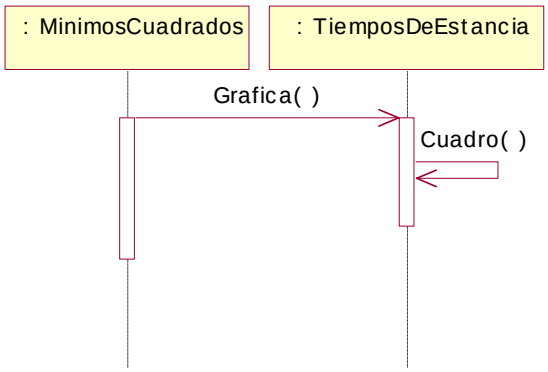
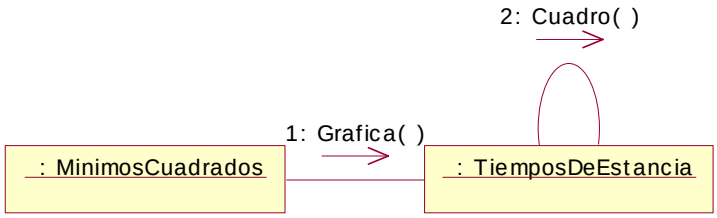


Diagrama de colaboración.



Especificación de Realización de Caso de Uso: Gráfica de Desgaste

Introducción

Propósito

El presente documento pretende mostrar de una manera clara y sencilla un panorama del caso de uso, mostrando sus clases y diagramas.

Alcance

El alcance de este Caso de Uso es el mostrar las principales interacciones entre las clases que lo componen, así como los mensajes que se transmiten para poder interrelacionarse.

Vista general

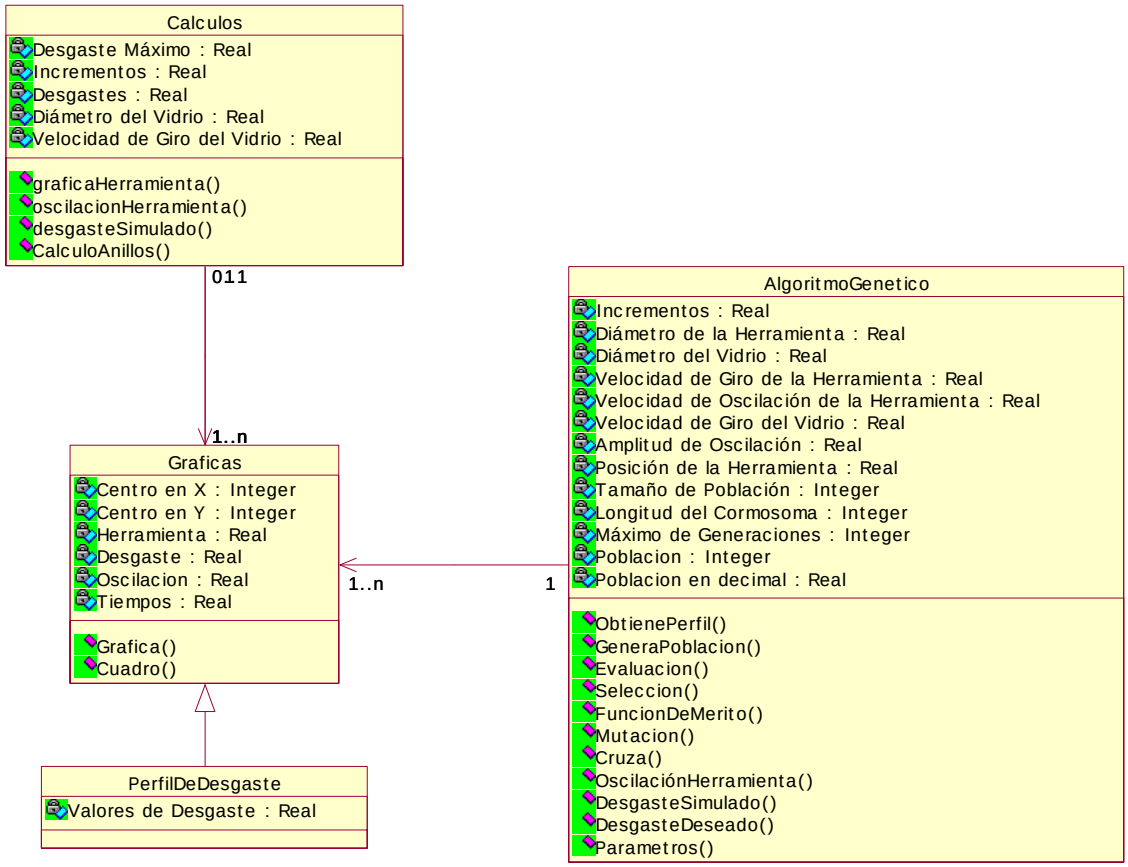
Se muestran las interacciones entre las clases, los mensajes que se envían y su secuencia.

Flujo de Eventos – Diseño

Flujo Básico

1. El programa muestra en pantalla la gráfica de Desgaste producida por la herramienta elegida de acuerdo a los parámetros ingresados por el Técnico de Pulido.

Diagrama de clases



Diagramas de secuencia

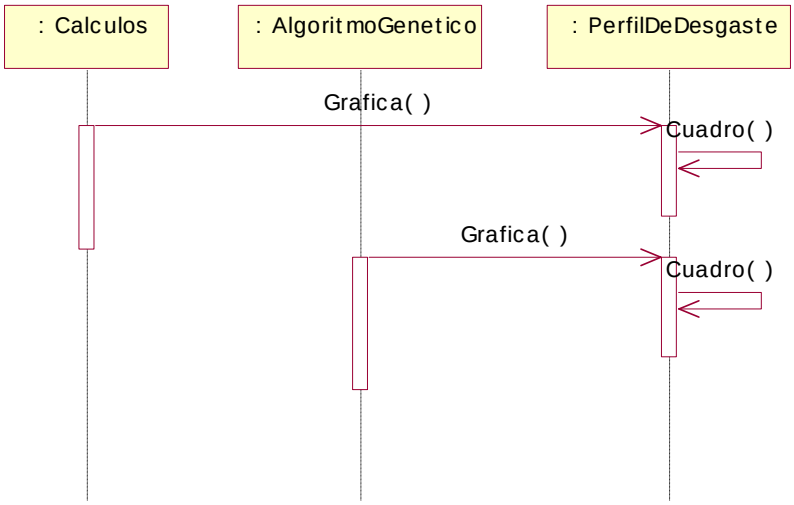
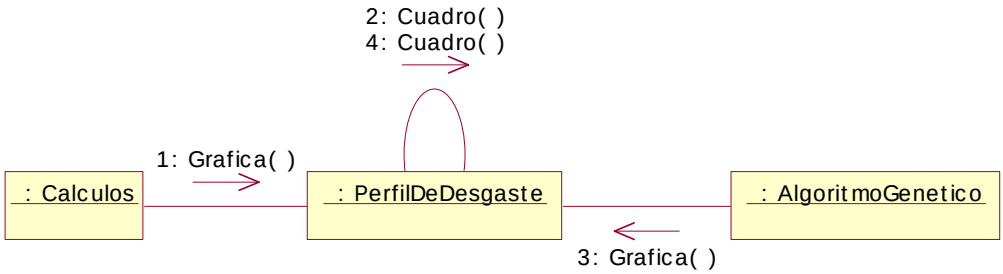


Diagrama de colaboración.



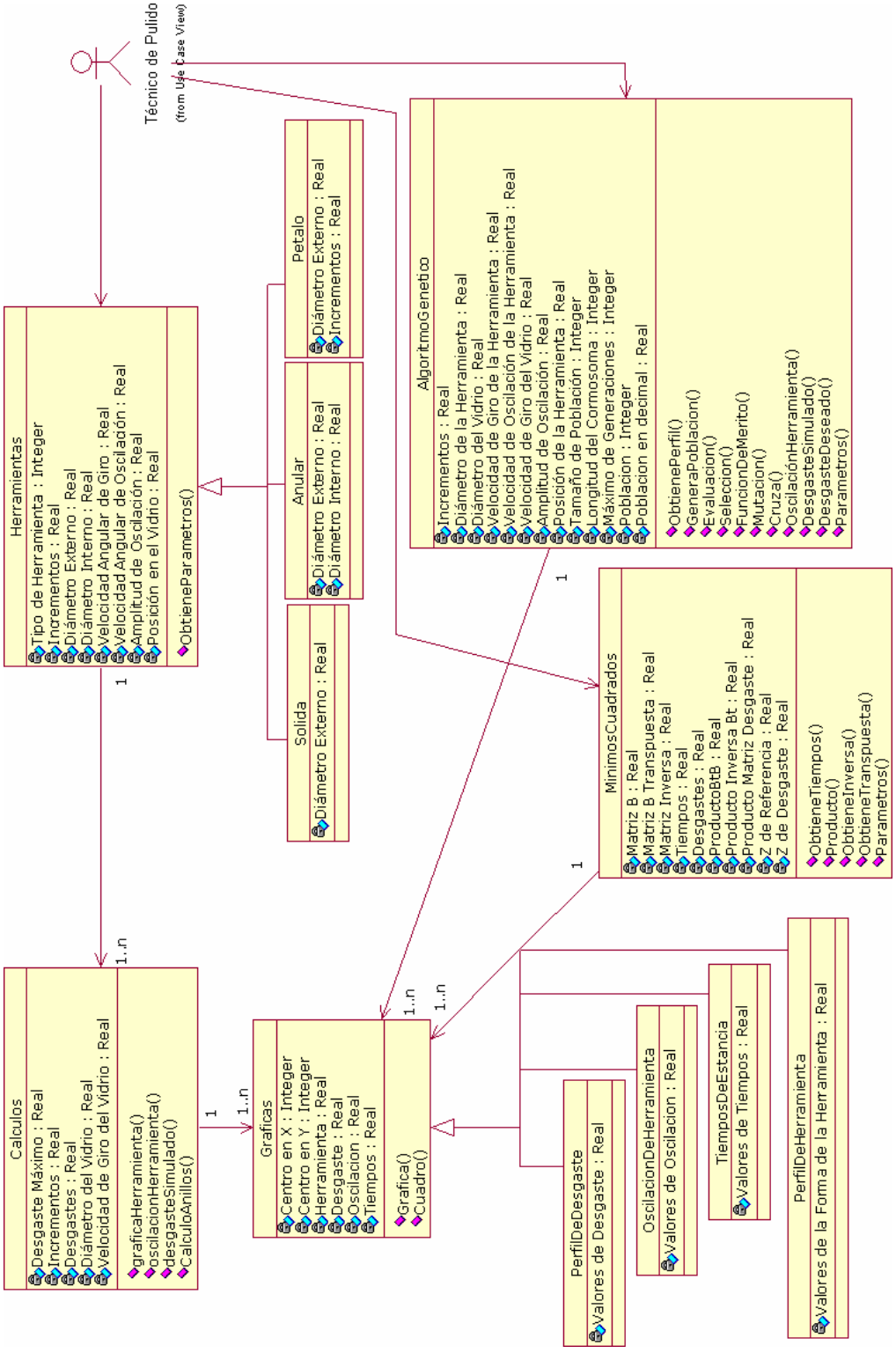
Apéndice D

Diagrama de Clases Unificado

Breve Descripción

El presente documento se muestran todas las clases que son utilizadas en el desarrollo del simulador así como la forma en que interactúan entre si.

Diagrama


Técnico de Pulido
(from Use Case View)
Herramientas
Tipo de Herramienta : Integer
Incrementos : Real
Diámetro Externo : Real
Diámetro Interno : Real
Velocidad Angular de Giro : Real
Velocidad Angular de Oscilación : Real
Amplitud de Oscilación : Real
Posición en el Vidrio : Real
ObtieneParametros()
Petalo
Diámetro Externo : Real
Incrementos : Real
Anular
Diámetro Externo : Real
Diámetro Interno : Real
Solida
Diámetro Externo : Real
AlgoritmoGenetico
Incrementos : Real
Diámetro de la Herramienta : Real
Diámetro del Vidrio : Real
Velocidad de Giro de la Herramienta : Real
Velocidad de Oscilación de la Herramienta : Real
Velocidad de Giro del Vidrio : Real
Amplitud de Oscilación : Real
Posición de la Herramienta : Real
Tamaño de Población : Integer
Longitud del Cormosoma : Integer
Máximo de Generaciones : Integer
Poblacion : Integer
Poblacion en decimal : Real
ObtieneParfil()
GeneraPoblacion()
Evaluacion()
Seleccion()
FuncionDeMerito()
Mutacion()
Cruza()
OscilaciónHerramienta()
DesgasteSimulado()
DesgasteDeseado()
Parametros()
MinimosCuadrados
Matriz B : Real
Matriz B Transpuesta : Real
Matriz Inversa : Real
Tiempos : Real
Desgastes : Real
ProductoBtB : Real
Producto Inversa Bt : Real
Producto Matriz Desgaste : Real
Z de Referencia : Real
Z de Desgaste : Real
ObtieneTiempos()
Producto()
ObtieneInversa()
ObtieneTranspuesta()
Parametros()
Calculos
Desgaste Máximo : Real
Incrementos : Real
Desgastes : Real
Diámetro del Vidrio : Real
Velocidad de Giro del Vidrio : Real
graficaHerramienta()
oscilacionHerramienta()
desgasteSimulado()
CalculoAnillos()
Graficas
Centro en X : Integer
Centro en Y : Integer
Herramienta : Real
Desgaste : Real
Oscilacion : Real
Tiempos : Real
Grafica()
Cuadro()
PerfilDeDesgaste
Valores de Desgaste : Real
OscilacionDeHerramienta
Valores de Oscilacion : Real
TiempoDeEstancia
Valores de Tiempos : Real
PerfilDeHerramienta
Valores de la Forma de la Herramienta : Real
1
1..n

Apéndice E

Diseño de la clase: Herramientas

Breve descripción

Este documento muestra el contenido de la clase Herramientas, sus operaciones y atributos que interactúan con las demás clases.

Responsabilidades

Esta clase se encarga de obtener del usuario el tipo de herramienta con la que va a trabajar y los parámetros correspondientes a dicha herramienta.

Relaciones

Esta clase está relacionada con la clase Calculos

Operaciones

ObtieneParametros( )

Ésta operación esta encargada de obtener los parámetros correspondientes al tipo de herramienta elegida por el usuario.

Atributos

Tipo de Herramienta

Contiene el tipo de herramienta elegida por el Técnico de Pulido.

Incrementos

Son variables con las cuales se calculan los tamaños angulares de los anillos incompletos que forman la herramienta de pétalo.

Diámetro Externo

Valor del diámetro externo de la herramienta seleccionada.

Diámetro Interno

Valor del diámetro interno de la herramienta seleccionada.

Velocidad Angular de Giro

Valor de la velocidad angular de giro de la herramienta elegida por el Técnico.

Velocidad Angular de Oscilación

Valor de la velocidad angular de oscilación de la herramienta elegida por el Técnico.

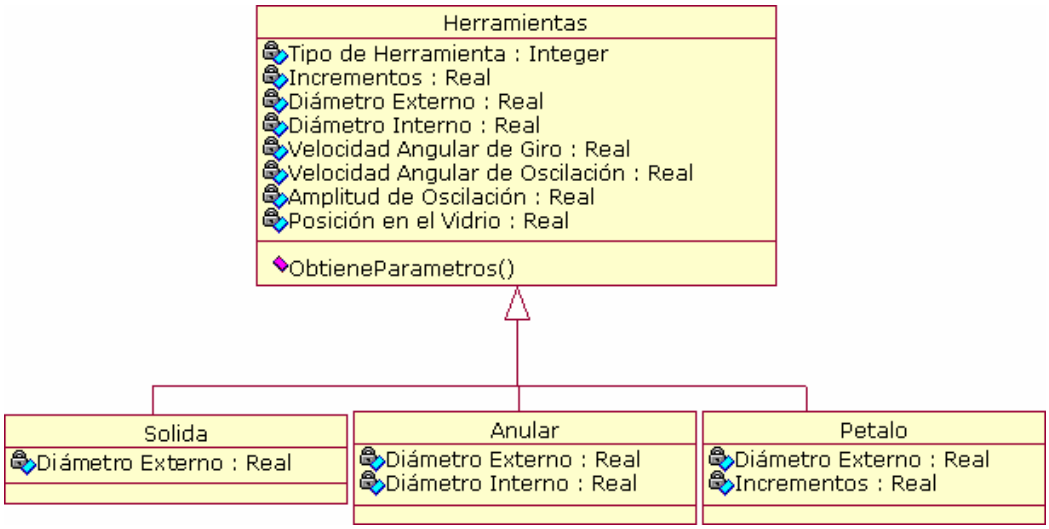
Amplitud de Oscilación

Es el valor de la amplitud que abarca la herramienta en el vidrio al oscilar en él.

Posición en el Vidrio

Contiene el valor de la posición de la herramienta sobre el vidrio, éste atributo define si la herramienta está centrada en el vidrio o no.

Diagrama



Diseño de la clase: Calculos

Breve descripción

Este documento muestra el contenido de la clase Calculos, sus operaciones y atributos que interactúan con las demás clases.

Responsabilidades

Esta clase tiene la responsabilidad de procesar todos los datos obtenidos del usuario y obtener en archivos de texto las posiciones x y y de las gráficas de la herramienta, desgaste y oscilación.

Relaciones

Esta clase está relacionada con la clase Herramientas y Graficas.

Operaciones

graficaHerramienta( )

Esta operación realiza los cálculos correspondientes para obtener la posición x y y de la herramienta seleccionada.

oscilacionHerramienta()

Esta operación realiza los cálculos correspondientes para obtener la posición x y y de la herramienta seleccionada oscilando sobre la superficie de vidrio.

desgasteSimulado()

Esta operación realiza los cálculos correspondientes para obtener la posición x y y del desgaste que genera la herramienta de acuerdo a los parámetros introducidos por el usuario.

CalculoAnillos()

Esta operación realiza los cálculos correspondientes para obtener la forma de la herramienta de acuerdo a los valores de los incrementos.

Atributos

Desgaste Máximo

Contiene el tipo de herramienta elegida por el Técnico de Pulido.

Incrementos

Es el tamaño de cada uno de los anillos incompletos de la herramienta de Pétalo.

Desgastes

Es un arreglo que contiene los desgastes generados en cada posición del vidrio por la herramienta.

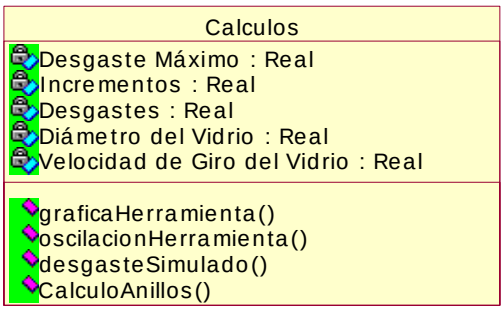
Diámetro del Vidrio

Valor del diámetro del vidrio.

Velocidad Angular de Giro del Vidrio

Valor de la velocidad angular de giro del vidrio.

Diagrama


Calculos
Desgaste Máximo : Real
Incrementos : Real
Desgastes : Real
Diámetro del Vidrio : Real
Velocidad de Giro del Vidrio : Real
graficaHerramienta()
oscilacionHerramienta()
desgasteSimulado()
CalculoAnillos()

Diseño de la clase: MinimosCuadrados

Breve descripción

Este documento muestra el contenido de la clase MinimosCuadrados, sus operaciones y atributos que interactúan con las demás clases.

Responsabilidades

Esta clase tiene la responsabilidad de obtener los tiempos en los que la herramienta sólida estará trabajando sobre la superficie de vidrio de acuerdo a los parámetros del vidrio.

Relaciones

Esta clase está relacionada con la clase Graficas.

Operaciones

ObtieneTiempos( )

Esta operación realiza los cálculos correspondientes para obtener los tiempos de estancia de la herramienta sólida sobre la superficie de vidrio, llamando en ésta operación a las operaciones que realizan los cálculos como Producto(), ObtieneInversa(), ObtieneTranspuesta().

Producto()

Adquiere el producto de dos matrices.

ObtieneInversa()

Obtiene la Inversa de una matriz.

ObtieneTranspuesta()

Genera la transpuesta de una matriz.

Parametros()

Adquiere las características del vidrio, dadas por el usuario.

Atributos

Matriz B

Matriz de funciones base.

Matriz B Transpuesta

Matriz de funciones base transpuesta

Matriz Inversa

Contiene la inversa del producto de matrices B^t B.

Tiempos

Valor de los tiempos de estancia obtenidos.

Desgastes

Matriz de Desgastes.

Producto BtB

Resultado del producto de la matriz B^tB

Producto Inversa Bt

Resultado del producto de la inversa de una matriz por la transpuesta.

Producto Matriz Desgaste

Resultado del producto de dos matrices.

Z de Referencia

Valores de sagitas en la esfera de referencia.

Z de Desgaste

Valores de sagitas en el desgaste deseado.

Diagrama

Diseño de la clase: AlgoritmoGenetico

Breve descripción

Este documento muestra el contenido de la clase AlgoritmoGenetico, sus operaciones y atributos que interactúan con las demás clases.

Responsabilidades

Esta clase tiene la responsabilidad de obtener la forma de la herramienta de pétalo de tal manera que se obtenga el desgaste deseado.

Relaciones

Esta clase está relacionada con la clase Graficas.

Operaciones

ObtienePerfil( )

Esta operación realiza los cálculos correspondientes para obtener la forma de la herramienta de pétalo.

GeneraPoblacion()

Genera número aleatorios que forman a la población inicial.

Evaluacion()

Evalúa la población.

Seleccion()

Hace el proceso de selección del algoritmo genético.

FuncionDeMerito()

Contiene el resultado de la función de mérito.

Mutacion()

Realiza el proceso de mutación dentro del algoritmo genético.

Cruza()

Realiza el proceso de cruza dentro del algoritmo genético.

OscilacionHerramienta()

Obtiene los puntos x y y de la herramienta oscilando sobre el vidrio.

DesgasteSimulado()

Obtiene el desgaste obtenido por la simulación y los ingresa en un archivo con las posiciones x y y para graficar.

DesgasteDeseado()

Obtiene el desgaste que se desea obtener dependiendo de las decisiones del usuario e ingresa éste desgaste en un archivo con las posiciones x y y para graficar.

Parametros()

Obtiene los parámetros correspondientes a la herramienta de pétalo.

Atributos

Incrementos

Valor del tamaño de los anillos que forman a la herramienta de pétalo.

Diámetro de la Herramienta

Valor del diámetro de la herramienta de pétalo.

Diámetro del Vidrio

Valor del diámetro del Vidrio.

Velocidad de Giro de la Herramienta

Valor de la velocidad angular de giro de la herramienta de pétalo.

Velocidad de Oscilación de la Herramienta

Valor de la velocidad angular de oscilación de la herramienta de pétalo.

Velocidad de Giro del Vidrio

Valor de la velocidad angular de giro del Vidrio.

Amplitud de Oscilación

El valor de la amplitud con la que oscila la herramienta sobre el vidrio.

Posición de la Herramienta

Contiene el valor de la posición de la herramienta sobre el vidrio, éste valor define si se encuentra trabajando en el centro del vidrio.

Tamaño de la Población

Valor del tamaño de la población del algoritmo genético.

Longitud del Cromosoma

Contiene la longitud del cromosoma del algoritmo genético.

Máximo de Generaciones

Valor del máximo de generaciones del algoritmo genético.

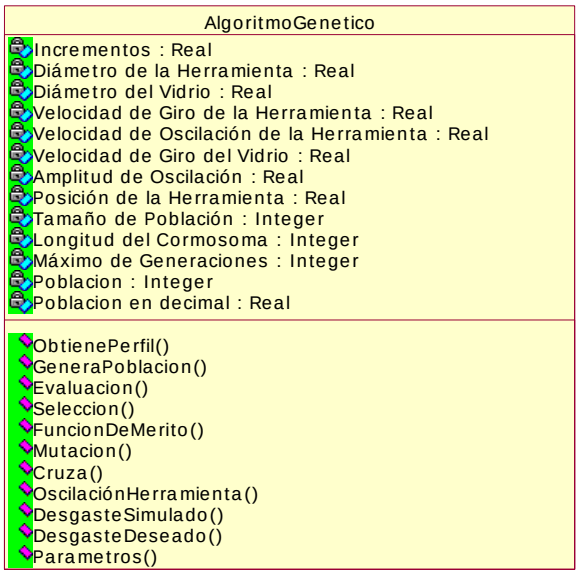
Población

Contiene a los individuos de la población.

Población en decimal

Contiene a los valores de los individuos en base decimal.

Diagrama



Diseño de la clase: Gráficas

Breve descripción

Este documento muestra el contenido de la clase Graficas, sus operaciones y atributos que interactúan con las demás clases.

Responsabilidades

Esta clase tiene la responsabilidad de procesar las posiciones x y y para mostrar en pantalla las diferentes gráficas de pulido.

Relaciones

Esta clase está relacionada con la clase Calculos, MinimosCuadrados y AlgoritmoGenetico.

Operaciones

Grafica ( )

Ésta operación grafica los datos obtenidos en pantalla.

Cuadro()

Dibuja un cuadro en pantalla para mostrar la gráfica.

Atributos

Centro de X

Valor del centro en el eje X de la pantalla.

Centro de Y

Valor del centro en el eje Y de la pantalla.

Herramienta

Contiene los valores que forman a la herramienta.

Desgaste

Contiene los valores que forman el perfil de desgaste.

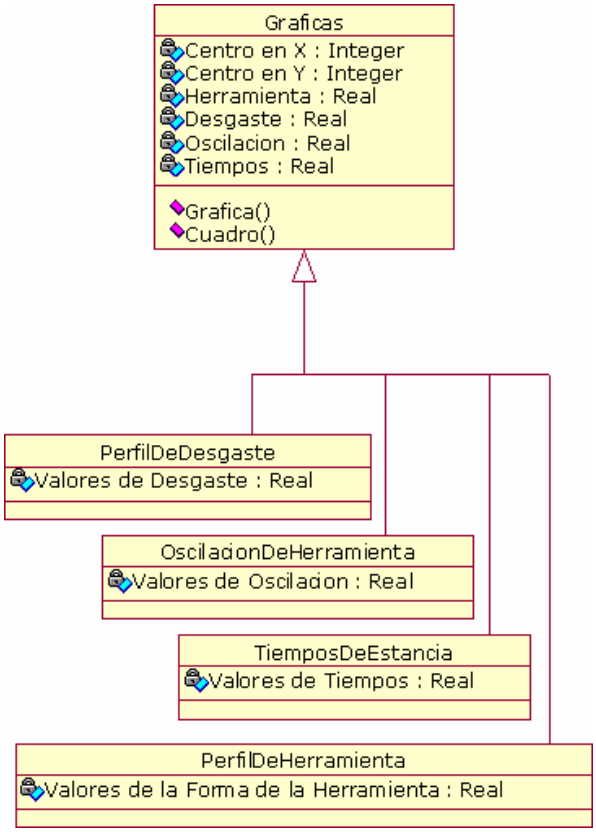
Oscilación

Contiene los valores que de oscilación de la herramienta.

Tiempos

Contiene los tiempos de estancia de la herramienta sólida.

Diagrama


Graficas
Centro en X : Integer
Centro en Y : Integer
Herramienta : Real
Desgaste : Real
Oscilacion : Real
Tiempos : Real
Grafica()
Cuadro()
PerfilDeDesgaste
Valores de Desgaste : Real
OscilacionDeHerramienta
Valores de Oscilacion : Real
TiemposDeEstancia
Valores de Tiempos : Real
PerfilDeHerramienta
Valores de la Forma de la Herramienta : Real

Apéndice F

Prueba de Usabilidad

Software para la Simulación del Proceso de Pulido utilizando el método de Pulido Clásico

Nombre del usuario:

Fecha:

Instrucciones generales

Estimado usuario:

Quisiéramos probar un nuevo software para la simulación del proceso de pulido utilizando el método de pulido clásico.

Le pedimos realizar unas tareas con este software. Describimos estas tareas en las siguientes páginas.

Lea cada tarea muy atentamente antes de comenzar.

Después de cada tarea llene el cuestionario correspondiente.

El facilitador le dirá cuándo puede avanzar a la siguiente tarea.

Estamos muy interesados en sus impresiones subjetivas y espontáneas. Es por eso que esperamos que manifieste sus pensamientos abiertamente durante la sesión (por ejemplo, lo que esperaba y no encontró en el software, lo que está bien o mal, etc.)

El facilitador está para responder sus dudas y preguntas durante la sesión.

En caso de que no tenga más preguntas, por favor comuníqueselo al facilitador para empezar con la tarea.

Tarea 1

Preguntas para Tarea 1

1) ¿Esta tarea fue fácil de hacer?

☐	Totalmente de acuerdo
☐	De acuerdo en gran parte
☐	Algo de acuerdo
☐	Algo en desacuerdo
☐	Totalmente en desacuerdo

2) La explicación de cada paso fue:

Buena	☐	☐	☐	☐	☐	Mala
Incomprensible	☐	☐	☐	☐	☐	Comprensible
Clara	☐	☐	☐	☐	☐	Confusa
Difícil	☐	☐	☐	☐	☐	Sencilla

3) ¿Fueron los pasos dentro de esta tarea claros?

☐	Muy claros
☐	Algo claros
☐	Poco claros
☐	No muy claros
☐	No fueron claros

4) ¿Qué tan satisfecho estuvo después de la tarea? ¿Qué cara corresponde a su estado de ánimo?

☐ ☐ ☐ ☐ ☐ ☐ ☐

5) ¿Ha tenido problemas con la tarea? ¿Qué fue particularmente bueno y malo? ¿Tiene alguna propuesta para mejorar el software?

6) ¿Es clara la información que representa el software?

Buena	☐	☐	☐	☐	☐	Mala
Incomprensible	☐	☐	☐	☐	☐	Comprensible
Clara	☐	☐	☐	☐	☐	Confusa
Difícil	☐	☐	☐	☐	☐	Sencilla

Si la respuesta fue negativa, ¿Por qué?

Preguntas finales

1. Si pudiera cambiar alguna parte del software. ¿Qué cambiaría? ¿Por qué?

2. Si conoce otro software disimulación, ¿Cómo lo compara con los resultados que presenta éste?

Prueba de Usabilidad

Software para la Simulación del Proceso de Pulido utilizando el método de Pulido Clásico

Nombre del observador:

Fecha:

Instrucciones generales

Estimado observador:

Quisiéramos probar un nuevo software para la simulación del proceso de pulido utilizando el método de pulido clásico.

Su trabajo es escuchar y observar al participante y al facilitador y tomar notas de todas las observaciones que le sean interesantes e importantes. Por ejemplo:

- ¿Se siente cómodo con el sistema el participante?
- ¿Se entienden el participante y el facilitador?
- ¿Hay situaciones en las cuales el participante tiene problemas para continuar con la tarea?
- ¿Está manifestando sus pensamientos y sentamientos el participante? Por favor tome notas
- Por favor esté atento a los tiempos que necesita el participante para cada tarea.

Después de la prueba le pedimos hacer un resumen con las observaciones más importantes (5 – 10).

Observaciones generales

Observaciones de tarea 1