

Desarrollo de software para modelado y fabricación de objetos de madera usando un robot SCARA

Roberto Loaeza Valerio

29 de abril de 2009

Índice general

| | |
|--|-----------|
| 1. Introducción | 6 |
| 1.1. Motivación | 7 |
| 1.2. Antecedentes | 7 |
| 1.3. Objetivos | 7 |
| 1.4. Alcances | 8 |
| 1.5. Metodología | 9 |
| 1.6. Contribuciones | 9 |
| 1.7. Descripción de los capítulos | 9 |
| | |
| 2. Ambiente de programación de robots manipuladores | 11 |
| 2.1. Robot | 11 |
| 2.1.1. Clasificación | 12 |
| 2.1.2. Estructura de los robots manipuladores | 12 |
| 2.1.3. Programación de tareas | 13 |
| 2.2. Revisión del Estado del Arte | 13 |
| 2.2.1. Programación de tareas | 14 |
| 2.2.2. Programación | 14 |
| 2.2.3. Desarrollo tecnológico | 15 |
| | |
| 3. Modelado de objetos | 18 |
| 3.1. Sistemas de coordenadas | 18 |
| 3.2. Primitivas de graficación | 19 |
| 3.2.1. Punto | 19 |
| 3.2.2. Linea | 19 |

| | | |
|-----------|--|-----------|
| 3.2.3. | Círculo | 19 |
| 3.2.4. | Elipse | 20 |
| 3.2.5. | Curva de Bezier | 20 |
| 3.2.6. | Elementos compuestos | 20 |
| 3.3. | Almacenamiento y recuperación | 20 |
| 3.3.1. | Almacenamiento en formato XML | 20 |
| 3.3.2. | Estructura de almacenamiento | 21 |
| 3.4. | Importacion de otros formatos | 22 |
| 3.4.1. | Imágenes PGM | 22 |
| 4. | Interfaces de software | 23 |
| 4.1. | Lenguaje de programación | 23 |
| 4.1.1. | Java | 24 |
| 4.2. | Diseño de la interfaz gráfica | 25 |
| 4.3. | Elección de sistema de renderización | 25 |
| 4.3.1. | OpenGL | 26 |
| 5. | Interfaz entre modelado y el robot SCARA | 28 |
| 5.1. | Robot SCARA | 28 |
| 5.1.1. | Características | 28 |
| 5.1.2. | Diseño | 28 |
| 5.1.3. | Limitantes | 28 |
| 5.2. | Conversión de modelado a acciones SCARA | 28 |
| 5.2.1. | Cinemática inversa | 29 |
| 5.3. | Conectividad PC-Robot | 31 |
| 6. | Pruebas | 32 |
| 6.1. | Introducción | 32 |
| 6.2. | Funcionalidad multiplataforma del software 3D | 32 |
| 6.2.1. | Windows | 32 |
| 6.2.2. | Linux | 32 |
| 6.3. | Conectividad multiplataforma entre software y el robot | 32 |

| | |
|--|-----------|
| 6.3.1. Windows | 32 |
| 6.3.2. Linux | 32 |
| 6.4. Pruebas en campo | 32 |
| 6.4.1. Realizacion de primitivas | 32 |
| 6.4.2. Imagenes PGM | 32 |
| 7. Conclusiones | 33 |
| 7.1. Conclusiones | 33 |
| 7.2. Trabajos futuros | 33 |
| Anexo | 35 |

Índice de figuras

| | | |
|------|--|----|
| 1.1. | Diagrama representativo del objetivo de la tesis | 8 |
| 2.1. | Categorías de los robots manipuladores y sus respectivas áreas de trabajo[PDF] . . . | 16 |
| 2.2. | Tipos de articulaciones: a) rotación b) prismática c) cilíndrica d) planar e) esférica . | 16 |
| 2.3. | Programación de un robot manipulador | 17 |
| 3.1. | Sistema de coordenadas de 3 dimensiones | 18 |
| 3.2. | Línea | 19 |
| 3.3. | Círculo | 20 |
| 3.4. | Elipse | 20 |
| 3.5. | Estructura de almacenamiento | 21 |
| 3.6. | PGM a Robot | 22 |
| 4.1. | Diseño de la interfaz gráfica | 25 |
| 4.2. | Aplicación utilizando OpenG y aceleración gráfica por hardware | 26 |
| 5.1. | Robot SCARA | 29 |
| 5.2. | Orientación del último eslabón | 30 |
| 5.3. | Ángulos β , ψ y Θ_1 | 31 |

Índice de tablas

| | |
|--|----|
| 1.1. Costos de adquisición de maquina industrial | 7 |
| 4.1. Lenguajes Sistemas Operativos | 24 |
| 4.2. Comparativa entre Direct3D, Java3D y OpenGL | 25 |

Capítulo 1

Introducción

En la actualidad las pequeñas y medianas empresas de la región maderera del estado de Michoacán, más preciso en la localidad de Paracho, requieren tener la totalidad de sus procesos automatizados para poder competir en el mercado tan saturado actualmente. Desafortunadamente la mayoría de estas empresas no cuentan con todos los procesos automatizados.

Los procesos que pueden automatizarse los dividiremos en dos categorías para fines de su estudio:

- Procesos administrativos
- Procesos industriales

Los procesos administrativos abarca todo proceso relacionado con documentación de la empresa, algunos ejemplos de estos procesos son los siguientes departamentos: recursos humanos, recursos financieros, ventas, compras, entre otros. Estos procesos o departamentos están parcial o totalmente cubiertos por aplicaciones ofrecidas por el gobierno de forma gratuita o haciendo uso de 7 aplicaciones de terceros, algunas de las aplicaciones utilizadas son:

- ContPAQ
- Compiere
- NomiPAQ
- SUA

Por otra parte los procesos industriales son todos los procesos en los que se manipula una materia prima y es transformada en un producto final mediante el uso de maquinaria industrial, y es en estos procesos donde se centra el trabajo de la presente tesis.

Teniendo las empresas la necesidad de contar con maquinaria industrial de precisión se ven en la necesidad de adquirir esta maquinaria en el extranjero debido a que en el país no existen empresas presten estos servicios.

| Empresa | Maquinaria | Costo de Adquisición | Costo de mantenimiento |
|---------|------------|----------------------|------------------------|
| PROCART | | | |
| | | | |
| | | | |
| | | | |

Tabla 1.1: Costos de adquisición de maquina industrial

Al usar maquinaria extranjera es evidente que los costos de compra, instalación, mantenimiento y programación son elevados, en la tabla 1.1 se muestran algunas maquinarias adquiridas por empresas de la localidad de Paracho Michoacán.

Observando la necesidad de la automatización en los procesos de fabricación de productos se optó por desarrollar una aplicación multiplataforma para el modelado de productos de madera haciendo uso de un robot SCARA realizado en la Facultad de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolas de Hidalgo.

1.1. Motivación

En la actualidad la mayoría de las pequeñas empresas de la región de Michoacán, en específico las de Paracho, no cuentan con procesos automatizados en la manufactura de sus materias primas.

Por otra parte, existen personas que tienen diseños muy innovadores pero por falta de una máquina que lo realice así como una aplicación para su diseño.

1.2. Antecedentes

Los antecedentes previos realizados en la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, de la Universidad Michoacana de San Nicolás de Hidalgo, es la construcción de un robot manipulador tipo SCARA de cuatro grados de libertad de plataforma abierta[robotumich], para el cual se desarrollará la aplicación de la presente tesis.

1.3. Objetivos

El objetivo general es realizar investigación en los aspectos computacionales relacionados con el modelado 3D de objetos físicos y su conversión en instrucciones que el robot SCARA pueda seguir para construir físicamente los modelos en madera, de forma que se desarrolle tecnología propia que promueva una industria robótica y se generen diversas aplicaciones que tengan un importante impacto social como económico.

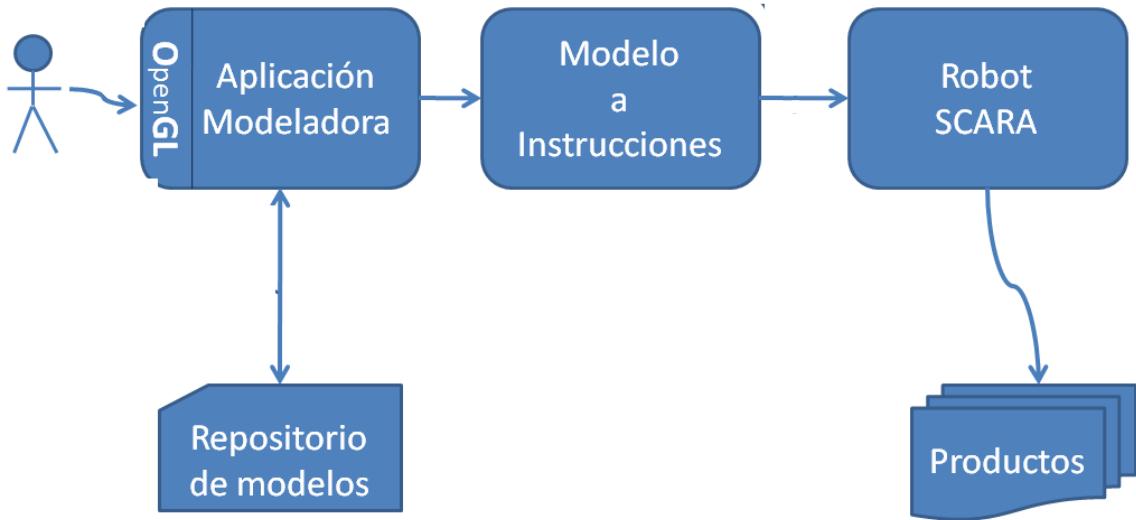


Figura 1.1: Diagrama representativo del objetivo de la tesis

La figura 1.1 representa gráficamente el objetivo, donde se puede apreciar que el presente proyecto de tesis tiene como entrada el diseño por parte de una persona el cual es interpretado para su futura producción en un robot SCARA.

Objetivos específicos

- Desarrollar una aplicación para diseñar objetos de madera.
- Desarrollar una aplicación para transferir el diseño elaborado hacia el robot SCARA.

1.4. Alcances

Los alcances previstos son:

- Implementación de algoritmos de gráfificación sobre la aplicación de diseño:
 - Puntos
 - Lineas
 - Círculos
 - Elipses
 - Curvas de Bezier

- Desarrollo de una aplicación multiplataforma(que pueda ejecutarse en la mayoría de los sistemas operativos actuales) capaz de modelar objetos de madera para su producción utilizando un robot SCARA.
- Desarrollo de interfaz entre la aplicación multiplataforma para modelar objetos de madera y el robot SCARA para la producción.

1.5. Metodología

Para cumplir los objetivos mencionados anteriormente se propone desarrollar una aplicación multiplataforma para el modelado de objetos de madera en el lenguaje de programación Java.

Para lograr una aplicación aceptable se desarrollarán prototipos de la aplicación hasta lograr una estabilidad considerable, la cual tenga una velocidad aceptable a la hora de generar los modelos en diferentes vistas.

Con el fin de mejorar el diseño se evaluará la aplicación continuamente en la empresa de Rafael Cardiel.

1.6. Contribuciones

A continuación se describen brevemente las contribuciones de este trabajo:

- Desarrollo de una aplicación multiplataforma para la construcción de modelos físicos de madera.
- Aplicación multiplataforma para la conexión entre el modelador y el robot SCARA.
- Código fuente, éste se distribuirá bajo la “*GNU General Public License v2*” y puede ser accedido en la siguiente url: modelando-madera.googlecode.com para su mejoramiento y/o ser tomado como base para futuros proyectos.

1.7. Descripción de los capítulos

En el capítulo 2 se presentarán una revisión del estado del arte asociado con la paquetería de software existente para el modelado/fabricación de objetos de madera.

En el capítulo 3 aborda las partes del software modelador tanto los algoritmos de graficación como la forma de interacción con el usuario final.

En el capítulo 4 se mostrarán las principales interfaces de software para la implementación de software 3D, se mostrarán pros y contras de cada una de estas y finalmente se detallará el funcionamiento de la interfaz OpenGL sobre la cual se realizará la implementación del software.

En el capítulo 5 trata las características del robot SCARA, la forma de conversión del modelo a acciones que el robot entienda así como un algoritmo de planeación para que el robot sea lo más óptimo posible. También se describirá la interfaz entre el software de modelado y el robot.

En el capítulo 6 se mostrarán resultados de pruebas realizadas en una primera instancia al software modelador, seguido de los resultados de la conectividad entre el software y el robot, y finalmente se verificará que tanta funcionalidad dote el algoritmo de planeación implementado.

En el capítulo 7 se aportarán las conclusiones generales resultado de la investigación abordada e ideas para trabajo de investigación posterior a realizar en el mismo campo del conocimiento.

Capítulo 2

Ambiente de programación de robots manipuladores

2.1. Robot

Un robot, es un agente artificial mecánico o virtual. Los primeros son maquinas usadas para realizar un trabajo automaticamente y son controlados por una computadora; por otro lado tenemos a los robots virtuales también conocidos como “bots” que son agentes virtuales de software.

En general para considerar a un agente artificial como robot, este debe cumplir algunas propiedades:

- Manipular cosas de su entorno.
- Sentir cosas del entorno.
- Contar con cierta inteligencia para tomar decisiones basadas en el ambiente o en una secuencia previamente programada.
- Ser programable
- Moverse en uno o más ejes.
- Realizar movimientos coordinados.

A continuación se presentan algunas definiciones realizadas por diferentes institutos:

- La ISO 8373 lo define como “*un manipulador multipropósito automáticamente controlado y reprogramable de mas de tres ejes*”.
- El Instituto de Robótica de América lo define como “*manipulador multifuncional y reprogramable diseñado para mover material, partes, herramientas o dispositivos especializados mediante varios movimientos programados para la realización de una variedad de tareas*”.

2.1.1. Clasificación

La Asociación Japonesa de Robots Industriales (JIRA, por sus siglas en inglés) clasificó los robots desde manipuladores simples hasta sistemas avanzados incorporando inteligencia artificial. A continuación se lista dicha clasificación la cual fue realizada a finales del siglo veinte[CER, p. 267]:

- Manipuladores operados manualmente. Máquinas que requieren ser operadas por humanos.
- Manipuladores secuenciales. Dispositivos que realizan una serie de tareas en la misma secuencia cada vez que son activados.
- Manipuladores programables. Se incluyen los tipos simples de los robots industriales.
- Robots controlados numéricamente. Servo robots.
- Robots con sensores. Robots que utilizan sensores de cualquier tipo, proximidad, presión, tactil, etc.
- Robots adaptativos. Robots que ajustan la forma en que trabajan para compensar cambios en el entorno.
- Robots inteligentes. Robots con controladores de salida sofisticados que pueden considerarse para procesar inteligencia artificial.
- Sistemas mecatrónicos inteligentes. Computadoras que controlan un conjunto de robots o dispositivos robóticos.

2.1.2. Estructura de los robots manipuladores

Básicamente la estructura de un robot manipulador es la de un brazo articulado. De manera más específica un robot manipulador industrial es una cadena cinemática abierta formada por un conjunto de eslabones interrelacionados mediante articulaciones las cuales permiten el movimiento de esta dicha cadena.

Área de trabajo

Tipos de articulaciones

Existen diferentes tipos de articulaciones como podemos observar en la figura 2.2:

- a) La articulación de rotación suministra un grado de libertad, es decir permite la rotación sobre el eje de la articulación.
- b) La articulación prismática consiste en una translación a lo largo del eje de la articulación.
- c) En la articulación cilíndrica existen dos grados de libertad, una rotación y una translación sobre el eje de la articulación.

- d) La articulación planar se caracteriza por el movimiento de desplazamiento en un plano, tiene dos grados de libertad.
- e) Por último la articulación esférica combina tres giros en tres direcciones perpendiculares en el espacio.

2.1.3. Programación de tareas

La secuencia que se tiene que realizar para programar una tarea en un robot manipulador generalmente sigue los pasos mostrados en la figura 2.3.

Software CAD

El software CAD(Diseño Asistido por Computadora, del inglés Computer Aided Design) es la aplicación donde el programador realiza un modelo/tarea que desea que realice el robot.

Software CAM

El software CAM(Manufactura Asistida por Computadora, del inglés Computer Aided Manufacturing) convierte el modelo/tarea en un código estandar para la mayoría del software de control CNC denominado código G.

Software CNC

El software CNC generalmente lee código G y realiza las operaciones de comunicación con el robot necesarias para convertirlas en el movimiento deseado.

2.2. Revisión del Estado del Arte

En la actualidad los robots son usados para realizar tareas peligrosas, difíciles, repetitivas y/o complicadas para los humanos. Esto usualmente toma la forma de un robot industrial usado en las líneas de producción. Algunas aplicaciones incluyen la limpieza de residuos tóxicos, minería y localización de minas terrestres.

En la actualidad los robots son usados para realizar una gran cantidad de tareas entre las cuales se encuentran las de realizar actividades peligrosas, difíciles, repetitivas y/o complicadas para los humanos.

Sin embargo la manufactura continúa siendo el principal mercado donde los robots son utilizados, en particular los robots articulados son los más usados comúnmente. Las aplicaciones incluyen soldado,

pintado y carga de maquinaria. En este ramo la industria automotriz ha tomado gran ventaja de esta nueva tecnología donde los robots han sido programados para reemplazar el trabajo de los humanos en muchas tareas repetitivas.

2.2.1. Programación de tareas

Software CAD

- AutoCAD
- SolidWorks
- RhinoCAD
- TurboCAD
- Graphite One CAD

Software CAM

- ArtCAM
- DeskCNC
- MeshCAM

Software CNC

- TurboCNC
- EMC
- DeskCNC

2.2.2. Programación

Un lenguaje de programación de robots sirve como interfaz entre el usuario humano y el robot industrial. Los manipuladores de robots se diferencian a sí mismos de la automatización fija por ser “flexibles”, es decir que son programables. No solo son programables los movimientos de los manipuladores sino que, a través del uso de sensores y comunicación con otros tipos de automatización, los manipuladores pueden adaptarse a las variaciones a medida que realizan su tarea.

Generalmente la programación de los robots se realiza fuera de linea, a continuación se listan paquetes existentes para programar:

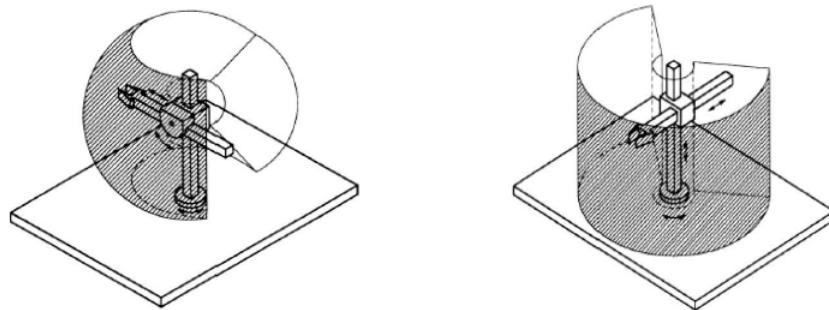
- MS Robotic Studio (~ \$ 4 000.00 M.N)

- Herramienta de programación visual para crear y depurar aplicaciones robóticas. El desarrollador puede interactuar con los robots mediante interfaces basadas en web o nativas al sistema operativo (MS Windows).
- Contiene simulación realística provista por el motor PhysX de AGEIA. Se posibilita la emulación por software o la aceleración por hardware.
- Se permiten varios lenguajes como: Microsoft Visual Studio Express languages (Visual C#® y Visual Basic® .NET), JScript® y Microsoft IronPython 1.0 Beta 1, y lenguajes de terceros que se adecuen a la arquitectura basada en servicios.
- Robots soportados: CoroWare's CoroBot (\$3 200), Lego Mindstorms NXT, iRobot Create y Robosoft's robots (38 a 65 K€).
- KUKA.OfficeLite (> \$55 000.00 USD)
 - Este sistema de programación posee las mismas características que el software de sistema KUKA: para el manejo y la programación se utiliza la interfaz de usuario Original KUKA y la sintaxis KRL: un lenguaje completo.
 - Disponibilidad de todo el repertorio de funciones de las respectivas ediciones del software de sistema. Sin embargo, no se pueden conectar dispositivos de hardware periféricos.
 - Comprobación de sintaxis mediante el compilador y el interpretador disponibles; creación de programas KRL de usuario ejecutables.
 - Control completo de la ejecución de un programa de aplicación de robot. Ello permite optimizar la duración de los ciclos.
 - El Techware de KUKA para optimización de programas se puede utilizar e instalar en todo momento. De este modo, en un PC estándar se puede disponer de todo el software de sistema Original, sin necesidad de emulaciones.
 - Las entradas originales se pueden simular.
 - KUKA.OfficeLite no se puede utilizar para controlar un robot.

2.2.3. Desarrollo tecnológico

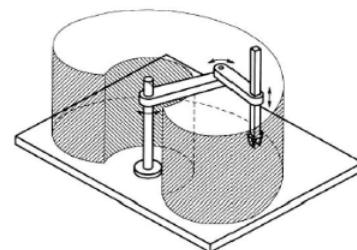
Desafortunadamente el desarrollo actual en el país está enfocado generalmente a robots móviles dejando a los manipuladores de lado.

Universidades como el Tecnológico de Monterrey [Dirección de Vinculación y Desarrollo campus Guadalajara] se está enfocando más en la planeación de movimientos para robots, así como el Instituto Tecnológico Autónomo de México [Laboratorio de robótica, ITAM] se enfocan al área de robots móviles pequeños.

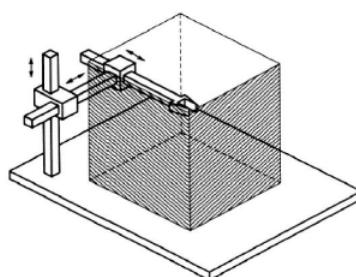


(a) Cilíndrico

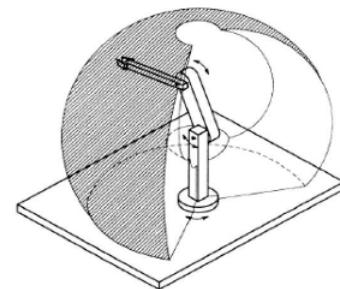
(b) Esférico



(c) S.C.A.R.A.



(d) Cartesiano



(e) Antropomórfico

Figura 2.1: Categorías de los robots manipuladores y sus respectivas áreas de trabajo[PDF]

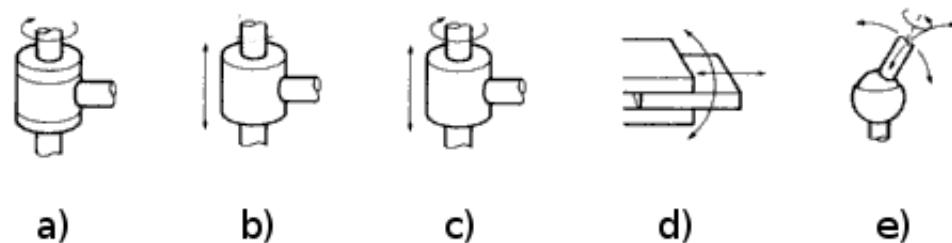


Figura 2.2: Tipos de articulaciones: a) rotación b) prismática c) cilíndrica d) planar e) esférica

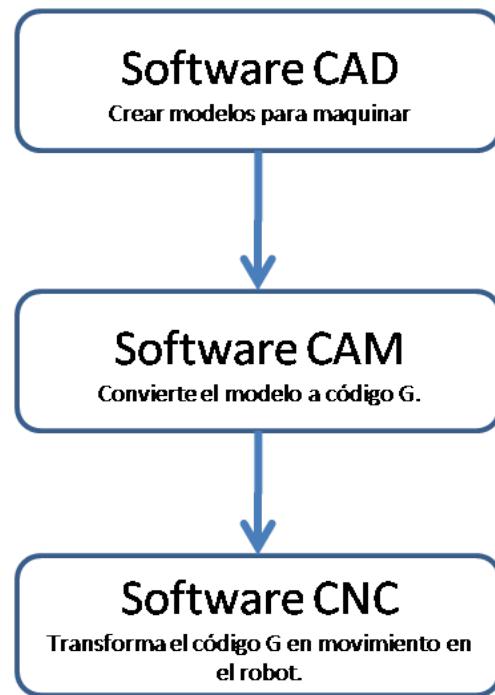


Figura 2.3: Programación de un robot manipulador

Capítulo 3

Modelado de objetos

Una aplicación de modelado de objetos proporciona una biblioteca de funciones que pueden utilizarse para crear diseños de objetos que posteriormente se pueden plasmar en madera o algún otro tipo de material que pueda ser moldeado por alguna de las herramientas soportadas por el robot. Estas funciones se denominan primitivas gráficas o simplemente primitivas.

Para describir un modelo, primero es necesario seleccionar un sistema de coordenadas cartesianas adecuado, que puede ser bidimensional o tridimensional. Después se describen los objetos del modelo proporcionando sus especificaciones geométricas. Por ejemplo se define una línea recta proporcionando la posición de los dos puntos extremos.

3.1. Sistemas de coordenadas

Un sistema de coordenadas es un conjunto de valores que permiten definir exactamente la posición de un punto cualesquiera en el espacio, debido a que se requiere poder realizar un modelo de un objeto real se requieren el sistema de coordenadas de tres dimensiones como se muestra en la figura 3.1.

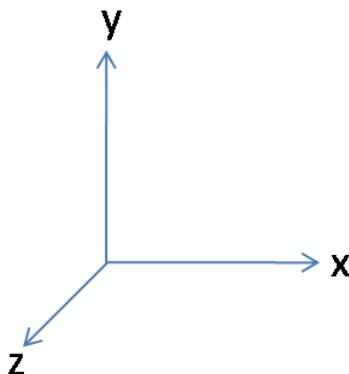


Figura 3.1: Sistema de coordenadas de 3 dimensiones

Sin embargo la aplicación desarrollada utiliza solo dos dimensiones al mismo tiempo para el desarrollo de modelos.

3.2. Primitivas de graficación

Las primitivas gráficas que describen la geometría de los objetos se denominan normalmente primitivas geométricas. Entre las primitivas geométricas más simples son las que indican posiciones de puntos y segmentos de líneas rectas. Adicionalmente se pueden incluir círculos, elipses y curvas tipo bezier.

3.2.1. Punto

El punto es el objeto más simple que puede representarse, solo se necesita un punto dentro del sistema de coordenadas.

$$\text{Punto } (x, y, z) \quad (3.1)$$

3.2.2. Línea

$$\text{Línea } (\text{Punto origen}, \text{Punto destino}) \quad (3.2)$$

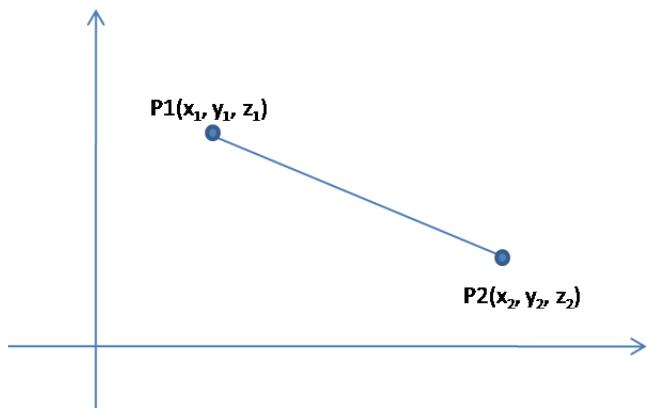


Figura 3.2: Línea

3.2.3. Círculo

$$\text{Círculo } (\text{Punto origen}, \text{radio}) \quad (3.3)$$

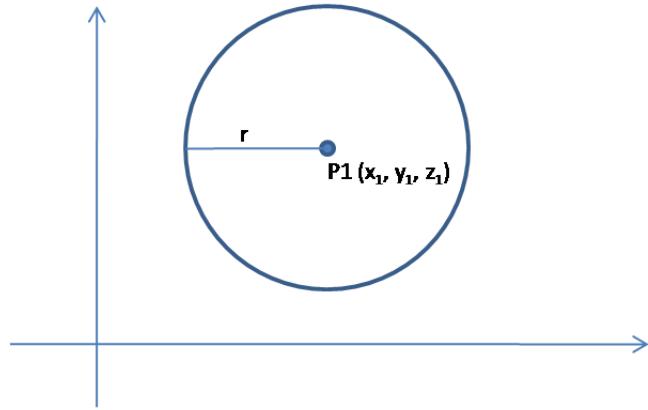


Figura 3.3: Círculo

3.2.4. Elipse

$$\text{Elipse} (\text{Punto origen}, \text{radio } a, \text{radio } b) \quad (3.4)$$

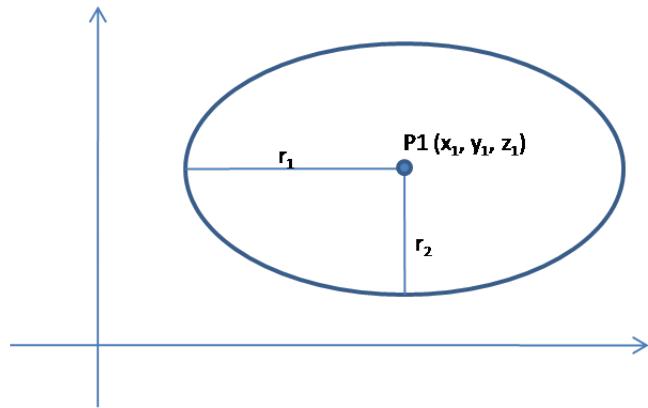


Figura 3.4: Elipse

3.2.5. Curva de Bezier

3.2.6. Elementos compuestos

3.3. Almacenamiento y recuperación

El almacenamiento y recuperación es una de las operaciones más importantes en un sistema,

3.3.1. Almacenamiento en formato XML

El formato XML

Algunas de las ventajas de este formato son:

- Independiente de la plataforma.
- Soporta código Unicode.
- El mismo documento define la estructura y los campos así como los valores respectivos.
- Es basado en estándares internacionales.

3.3.2. Estructura de almacenamiento

La estructura de almacenamiento puede observarse en la figura 3.5. Como puede observar la estructura es simple, compacta y fácil de entender.

```

<modelador>
  <fig tipo="1">
    <x0> val_x0 </x0>
    <y0> val_y0 </y0>
    <z0> val_z0 </z0>

    . . .
    . . .

    <xN> val_xN </xN>
    <yN> val_yN </yN>
    <zN> val_zN </zN>
  </fig>
  . . .
  <fig tipo="M">
  . . .
</modelador>

```

Figura 3.5: Estructura de almacenamiento

En la figura 3.5 puede observarse:

- No existe límite en la cantidad de primitivas usadas en un diseño.
- Los puntos de control de una primitiva pueden no tener límite, es decir una primitiva puede tener N puntos de control.
- Cada primitiva es identificada mediante mediante un número entero (tipo):
 - Punto (1)
 - Línea (2)
 - Poli-Línea (3)
 - Círculo (4)

- Elipse (5)
- Curva de Bezier (6)

3.4. Importacion de otros formatos

3.4.1. Imágenes PGM

Una imagen PGM es la representacion a escala de grises de una imagen.

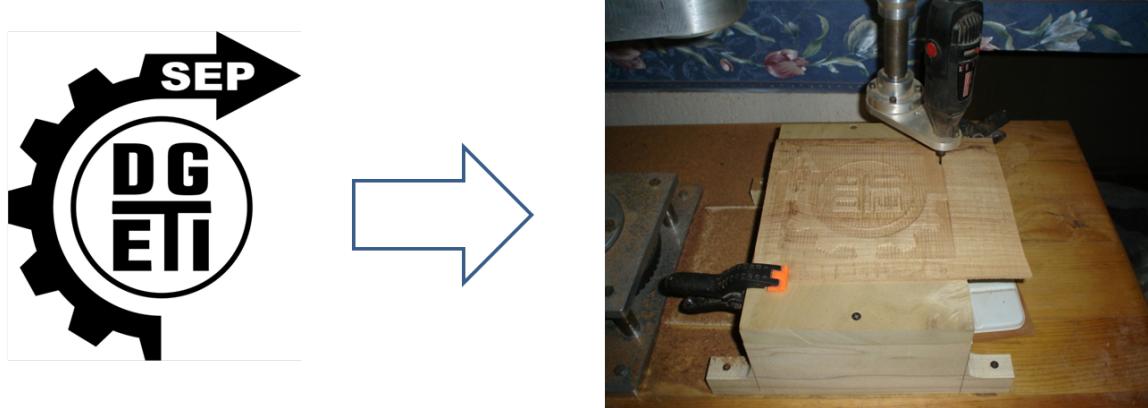


Figura 3.6: PGM a Robot

Capítulo 4

Interfaces de software

La interfaz de usuario es uno de los apartados con más relevancia de un sistema de computo, en la actualidad todo sistema debe contener una herramienta con la que el usuario pueda ordenar al computador que hacer.

Una interfaz de usuario mal diseñada puede causar que el mejor sistema de cómputo sea ineficaz y por lo tanto sea deshechado, es por esto que se debe realizar un buen diseño de la interfaz de usuario, la cual incluye:

- Elección de lenguaje de programación.
- Diseño de la interfaz gráfica.
- Elección de sistema de renderización.

4.1. Lenguaje de programación

En la actualidad existe una gran cantidad de lenguajes de programación, la mayoría de propósito general y el resto de propósito específico, a continuación se listan los más populares de ambas categorías:

- C.
- Basic.
- Java.
- .Net.
- Perl.
- PHP.

Estos podrían ser los más utilizados hoy por hoy, pero debido a la heterogeneidad de sistemas operativos y arquitecturas, no todos los lenguajes de programación son aplicables.

En el caso de los lenguajes compilados es poco probable su funcionalidad en diferentes plataformas debido a la dependencia de tipo de procesador que adquieren al momento de compilar, por ejemplo si se compila en un procesador tipo RISC es imposible ejecutarlo en un procesador tipo CISC.

Por otro lado los lenguajes interpretados son muy portables a todas las arquitecturas y sistemas operativos debido a que no dependen de un tipo de arquitectura pero requieren de que exista un programa nativo que los interprete, es decir, que si la empresa desarrolladora del lenguaje interpretado no libera una versión de su intérprete para un sistema operativo en una arquitectura determinada los programas escritos en este lenguaje no se ejecutarán.

| | Windows | Linux | MacOS |
|-------|---------|-------|-------|
| C | X | X | X |
| Basic | X | | |
| Java | X | X | X |
| .Net | X | X | |
| Perl | X | X | X |
| PHP | X | X | |

Tabla 4.1: Lenguajes Sistemas Operativos

En la tabla 4.1 se puede apreciar los alcances de los lenguajes de programación más desarrollados y más difundidos en la actualidad, dejando claro que Java es el lenguaje de programación con mayor portabilidad debido a que se cuenta con intérpretes para todas las arquitecturas y además que no se requiere modificar el código fuente para ejecutarse en éstas.

4.1.1. Java

Java es una plataforma de software desarrollada por Sun Microsystem, de forma que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales.

Esta plataforma está compuesta por:

- El Lenguaje de programación.
- La máquina virtual de Java (JVM).
- Un conjunto de bibliotecas estándar, conocidas como API.

El lenguaje de programación usa la sintaxis de C++, incorpora sincronización, manejo de tareas e interfaces como un mecanismo alternativo a la herencia múltiple de C++.

La máquina virtual de Java (JVM) es un programa nativo (un ejecutable de una plataforma específica) capaz de interpretar y ejecutar código binario especial (llamado Java Bytecode), el cual es generado a partir del compilador de Java. Es por esto que Sun Microsystem ha liberado versiones de

su JVM para las arquitecturas y sistemas operativos más utilizados, volviendo así a las aplicaciones desarrolladas en Java multiplataforma.

4.2. Diseño de la interfaz gráfica

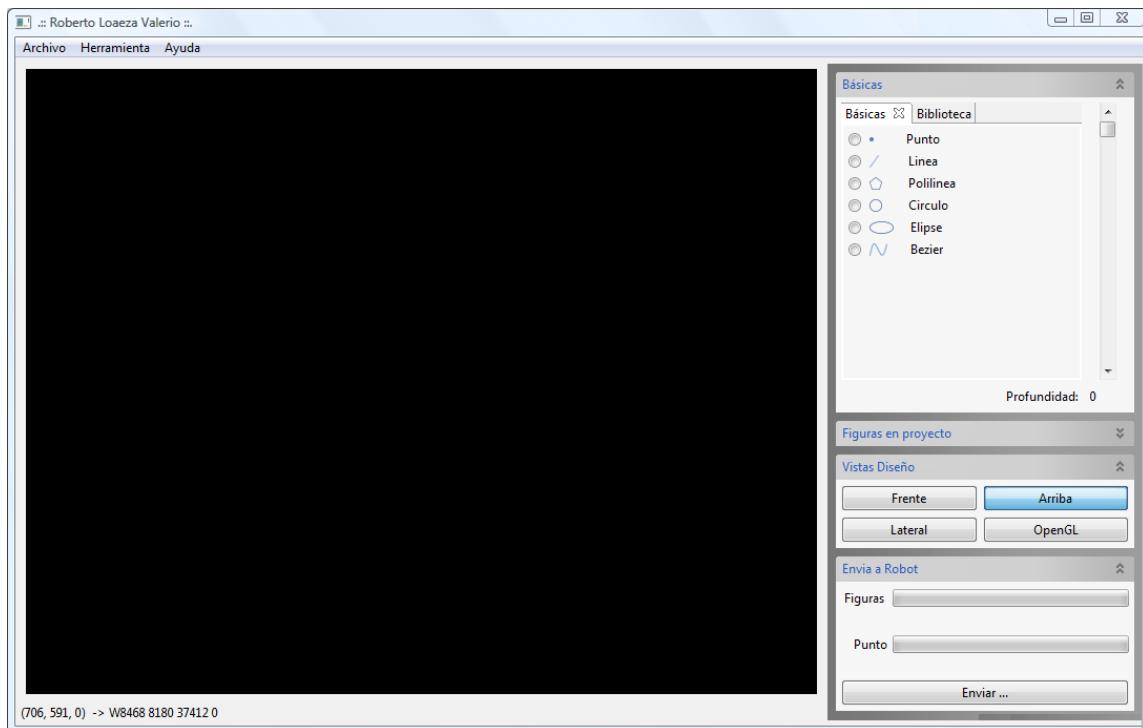


Figura 4.1: Diseño de la interfaz gráfica

4.3. Elección de sistema de renderización.

La elección debe realizarse tomando en cuenta ventajas y desventajas de las tecnologías disponibles las cuales se muestran en la tabla 4.2.

| | Direct3D | Java3D | OpenGL |
|------------------------|----------|--------------------|-----------------|
| Plataformas soportadas | Windows | Multiplataforma | Multiplataforma |
| Soporte nativo | Si | No | Si |
| Licencia | EULA | GNU General Public | SGI |

Tabla 4.2: Comparativa entre Direct3D, Java3D y OpenGL

Debido a que Direct3D solo soporta la plataforma MS-Windows fue desechada y por otra parte Java3D tiene una dependencia con OpenGL o con Direct3D también fue desechada dejando como sistema de renderización a OpenGL.

4.3.1. OpenGL

OpenGL es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics Inc. (SGI) en 1992 .

Las funciones y comandos de esta API están integrados en las tarjetas de aceleración gráfica 3D, o en su defecto son emuladas por el sistema operativo. Es claro que para el propósito específico de modelado 3D no se debe realizar emulación por software sino se debe tener acceso directo al hardware.

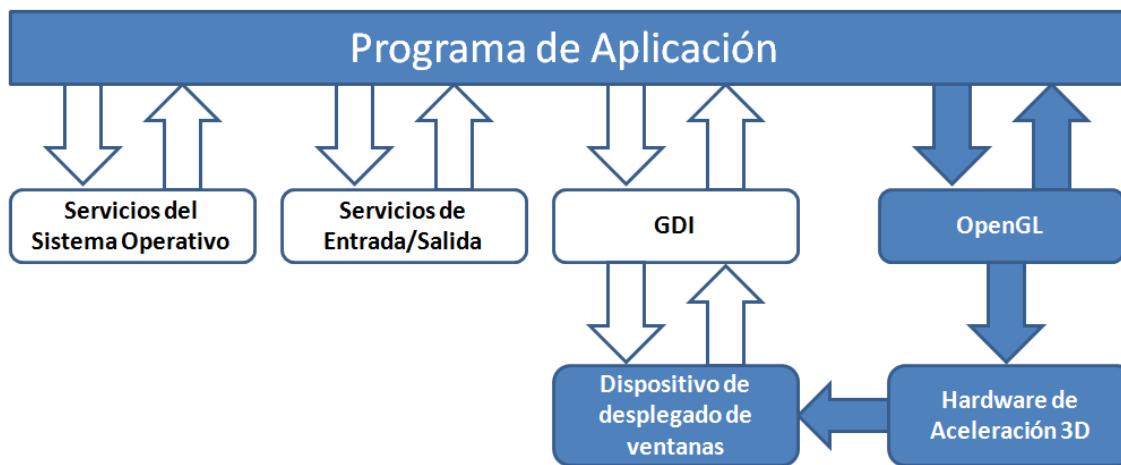


Figura 4.2: Aplicación utilizando OpenG y aceleración gráfica por hardware

En la figura 4.2 se muestra una aplicación utilizando OpenGL y aceleración gráfica por hardware; se puede observar que las llamadas a la interfaz de aplicación OpenGL pasan directamente al hardware de aceleración y no pasan a través de la interfaz de dispositivo de gráfico (GDI) que sería el caso si se tratara de simulación por software.

GLUT

Es un conjunto de herramientas OpenGL, un sistema de ventanas independientes para escribir programas OpenGL en lenguajes C y FORTRAN. Implementa un simple interfaz de programación para aplicaciones (API) para OpenGL. Por su parte GLUT hace considerablemente fácil la programación OpenGL.

GLUT provee un soporte para las plataformas Windows, Linux, Mac. Con la única desventaja de tener que recompilar para ser usado en otra plataforma.

JOGL (JSR-231)

Jogl es un enlace del lenguaje de programación Java para la API de OpenGL. Soporta integración con los estándares de Java AWT y Swing. Jogl provee acceso a las más recientes rutinas de OpenGL (OpenGL 2.0 con extenciones de fabricante) así como una plataforma independiente para accesar a hardware acelerador.

Jogl tambien prvee algunas de las mas populares características introducidas por otros enlaces existentes para OpenGL como son GL4Java, LWJGL, GLUT entre otros.

Es importante mencionar que esta tecnologia es código abierto y fue iniciada por Game Technology Group.

Capítulo 5

Interfaz entre modelado y el robot SCARA

La interfaz entre el software modelador y el robot es una parte muy importante, ya que si esta llega a colapsar o malfuncionar se verá reflejado en un trabajo final no satisfactorio.

5.1. Robot SCARA

El robot con el cual se trabajó se puede apreciar en la figura 5.1. Este robot cuenta con 4 grados de libertad construido en el departamento de robótica de la FIE, fijo en su base y hecho de acero y aluminio

5.1.1. Características

Entre las características más relevantes acerca del robot se tiene:

- 4 Grados de libertad.
- Alcance máximo de su brazo 50 cm.
- Construido en acero y aluminio.
- Peso alrededor de 45 Kg.

5.1.2. Diseño

5.1.3. Limitantes

5.2. Conversión de modelado a acciones SCARA

Debido a que en el modelo solo contamos con puntos definidos dentro del sistema de coordenadas cartesianas de tres dimensiones es necesario calcular el conjunto de ángulos de las articulaciones

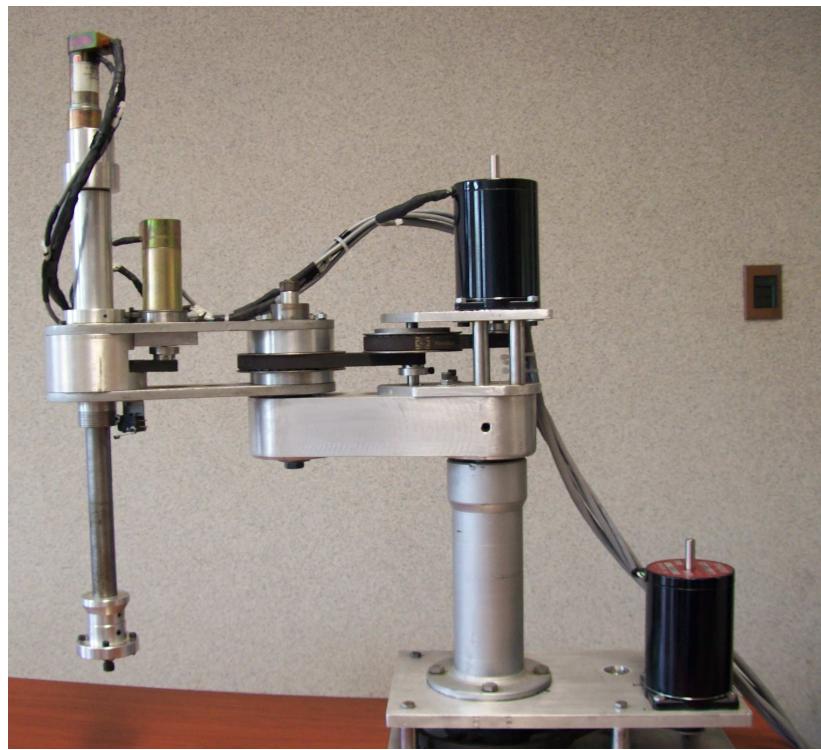


Figura 5.1: Robot SCARA

para lograr el resultado deseado. Esto lo solucionaremos usando cinemática inversa.

5.2.1. Cinemática inversa

En la cinemática inversa se conoce la posición y la orientación del elemento terminal referido a la base y se desea determinar los angulos articulares para alcanzar dicha posición.

Existen varias soluciones a este problema:

- Soluciones Cerradas(analíticas)
 - Solución algebraica
 - Solución geométrica
- Soluciones Numéricas
 - Iterativas

Debido a su naturaleza iterativa, las soluciones numéricas son generalmente mucho más lentas que las solución de forma cerrada, en algunos casos las soluciones numéricas son tan lentas que pueden ocasionar problemas de cinemática es por esto que solo nos enfocaremos en las soluciones analíticas.

Solución Algebraica

Solución Geométrica

Recordando algunas identidades trigonométricas:

$$\frac{\sin(A)}{a} = \frac{\sin(B)}{b} = \frac{\sin(C)}{c} \quad (5.1)$$

$$a^2 = b^2 + c^2 - (2bc) \cos(A) \quad (5.2)$$

Primero se sabe que la orientación del ultimo eslabón es la suma de las variables articulares, como se muestra en la figura 5.2.

$$\Theta = \Theta_1 + \Theta_2 + \Theta_3 \quad (5.3)$$

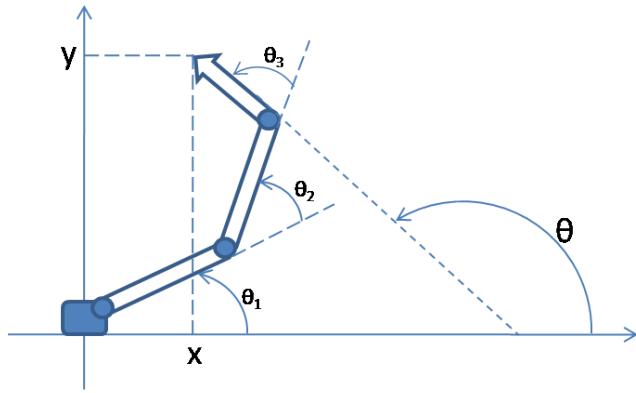


Figura 5.2: Orientación del ultimo eslabon

Se calcula θ_2 aplicando ec. 5.2:

$$x^2 + y^2 = l_1^2 + l_2^2 - (2l_1l_2) \cos(180 - \Theta_2) \quad (5.4)$$

debido a que:

$$\cos(180 - \Theta_2) = -\cos(\Theta_2) \quad (5.5)$$

Resulta:

$$\cos(\Theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \quad (5.6)$$

Se calcula θ_1 :

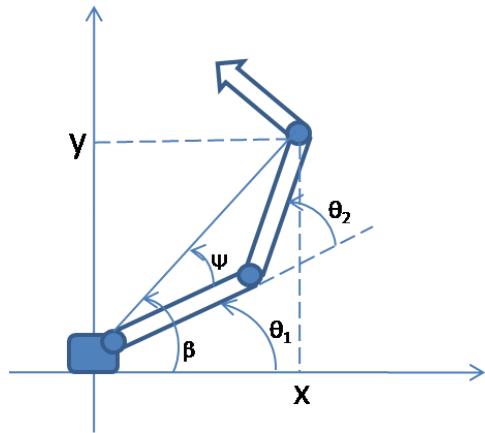


Figura 5.3: Ángulos β , ψ y Θ_1

Si se definen dos ángulos como se muestra en la figura 5.3 se cumple:

$$\Theta_1 = \beta - \psi \quad (5.7)$$

El ángulo β se calcula:

$$\sin(\beta) = \frac{y}{\sqrt{x^2 + y^2}} \quad (5.8)$$

y para el ángulo ψ se usa ec. 5.1 y se tiene:

$$\cos(\psi) = \frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}} \quad (5.9)$$

Finalmente se calcula θ_3 usando la siguiente ecuación:

$$\Theta_3 = \Theta - \Theta_1 - \Theta_2 \quad (5.10)$$

5.3. Conectividad PC-Robot

Capítulo 6

Pruebas

6.1. Introducción

6.2. Funcionalidad multiplataforma del software 3D

6.2.1. Windows

6.2.2. Linux

6.3. Conectividad multiplataforma entre software y el robot

6.3.1. Windows

6.3.2. Linux

6.4. Pruebas en campo

6.4.1. Realizacion de primitivas

6.4.2. Imagenes PGM

Capítulo 7

Conclusiones

7.1. Conclusiones

7.2. Trabajos futuros

Anexo Glosario

CISC Complex-Intruction-Set Computing.

CRM Customer Relationship Management.

ERP Enterprise Resource Planning.

ISO International Organization for Standardization.

JIRA Japan Industrial Robot Association.

OpenGL Open Graphics Library.

PGM Portable GrayMap.

RISC Reduced-Intruction-Set Computing.

SCARA Selective Compliant Articulated Robot Arm.

XML Extensible Markup Language.

Bibliografía

- [WIKI-ROBOT] http://en.wikipedia.org/wiki/Industrial_robot
- [GLUT] <http://www.opengl.org/resources/libraries/glut/>
- [JOGL] <https://jogl.dev.java.net/>
- [RAH] Thomas R. Kurfess, Robotics and Automation Handbook, E.U.A.: ed. CRC PRESS, 2005.
- [CER] Stan Gibilisco, Concise Encyclopedia of Robotics, E.U.A.: ed. McGraw-Hill, 2003
- [PJGD] Andrew Davison, Pro Java 6 3D Game Development Java 3D, JOGL, JInput and JOAL APIs, E.U.A.: ed. Apress, 2007
- [RMRM] Aníbal Ollero Baturone, Robótica: Manipuladores y robots móviles, Marcombo Editorial.
- [1] Wilbert O. Galitz, The Essential Guide to User Interface Design, E.U.A.: ed.: Wiley, 2007.
- [robotumich] Robot michoacan