

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

Roberto Loaeza Valerio

9 de mayo 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. Estructura de los robots manipuladores	12
2.1.2. Clasificación	14
2.1.3. Programación de tareas	14
2.2. Revisión del Estado del Arte	15
2.2.1. Programación de tareas	16
2.2.2. Desarrollo tecnológico	18
3. Modelado de objetos	19
3.1. Sistemas de coordenadas	19
3.2. Primitivas de graficación	20
3.2.1. Punto	20
3.2.2. Linea	20
3.2.3. Círculo	21

3.2.4. Elipse	21
3.2.5. Curva de Bezier	22
3.2.6. Elementos compuestos	22
3.3. Transformaciones	23
3.3.1. Escalado	23
3.3.2. Traslación	23
3.3.3. Rotación	24
3.4. Almacenamiento y recuperación	24
3.4.1. Almacenamiento en formato XML	25
3.4.2. Estructura de almacenamiento	25
3.5. Importación de otros formatos	26
3.5.1. Imágenes PGM	26
4. Interfaz de software	27
4.1. Diseño de la interfaz de usuario	27
4.1.1. Principios generales de un buen diseño	27
4.2. Lenguaje de programación	29
4.2.1. Java	31
4.3. Elección de sistema de renderización.	31
4.3.1. OpenGL	32
5. Interfaz entre modelado y el robot SCARA	34
5.1. Robot SCARA	34
5.1.1. Características	34
5.1.2. Arquitectura	35
5.1.3. Limitantes	36
5.2. Conversión de modelado a acciones SCARA	36
5.2.1. Cinemática inversa	36
5.3. Conectividad PC-Robot	38

6. Pruebas	40
6.1. Introducción	40
6.2. Funcionalidad multiplataforma del software 3D	40
6.2.1. Windows	40
6.2.2. Linux	40
6.3. Conectividad multiplataforma entre software y el robot	40
6.3.1. Windows	40
6.3.2. Linux	40
6.4. Pruebas en campo	40
6.4.1. Realizacion de primitivas	40
6.4.2. Imagenes PGM	40
7. Conclusiones	41
7.1. Conclusiones	41
7.2. Trabajos futuros	41
Comandos soportados por el robot SCARA	43
Anexo	45

Índice de figuras

1.1.	Diagrama representativo del objetivo de la tesis	8
2.1.	Tipos de articulaciones: a) rotación b) prismática c) cilíndrica d) planar e) esférica .	13
2.2.	Categorías de los robots manipuladores y sus respectivas áreas de trabajo[Sanchez07].	15
2.3.	Programación de un robot manipulador.	16
3.1.	Sistema de coordenadas de 3 dimensiones	19
3.2.	Línea	21
3.3.	Círculo	21
3.4.	Elipse	22
3.5.	Elemento compuesto	22
3.6.	Escalamiento	23
3.7.	Traslación	24
3.8.	Rotación	25
3.9.	Estructura de almacenamiento	26
3.10.	PGM a Robot	26
4.1.	Diseño de la interfaz gráfica	30
4.2.	Aplicación utilizando OpenG y aceleración gráfica por hardware	32
5.1.	Robot SCARA	35
5.2.	Módulo Adapt9S12DP256 del microcontrolador 68hc12[Rodriguez08]	36
5.3.	Orientación del último eslabón	37
5.4.	Ángulos β , ψ y Θ_1	38

Índice de tablas

1.1.	Costos de adquisición de maquina industrial	7
4.1.	Lenguajes Sistemas Operativos	31
4.2.	Comparativa entre Direct3D, Java3D y OpenGL	31
5.1.	Bibliotecas para comuniación RS232	39

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 abarcan 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 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, 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 que 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, construido en la Facultad de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolas de Hidalgo.

1.1. Motivación

Dos han sido las razones que motivaron la elaboración de la presente tesis:

En la actualidad la mayoría de las pequeñas empresas de las regiones de Michoacán, en particular 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 recursos económicos para adquirir una máquina industrial que lo realize, así como una aplicación para su diseño.

La segunda razón es la que más me ha impulsado a trabajar en este campo, toda vez que nos permite aplicar los conocimientos adquiridos en la Maestría y la oportunidad de aportar la solución tecnológica a estos casos.

1.2. Antecedentes

El antecedente más cercano que se tiene, es el trabajo realizado en la División de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica, de la Universidad Michoacana de San Nicolás de Hidalgo, que es la construcción de un robot manipulador tipo SCARA, de cuatro grados de libertad de plataforma abierta[Rodriguez08], 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 y económico.

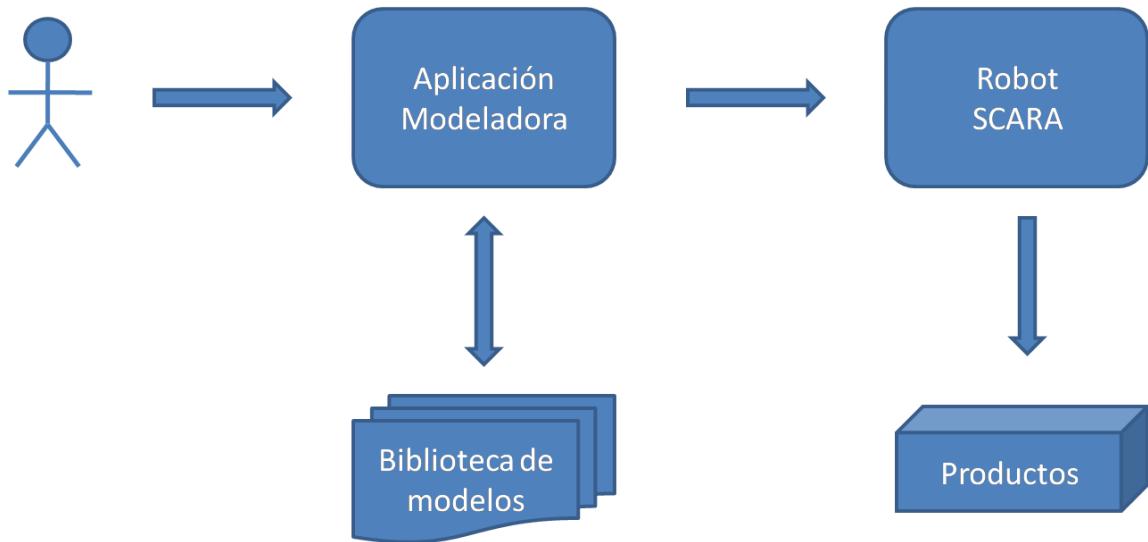


Figura 1.1: Diagrama representativo del objetivo de la tesis

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 gráficación 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 señor Rafael Cardiel, domiciliado en esta población.

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 presentará una revisión del estado del arte asociado con la paquetería de software existente para el modelado/fabricación de objetos de madera.

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 éstas y finalmente se detallará el funcionamiento de la interfaz realizada.

El capítulo 5 trata las características del robot SCARA, la forma de conversión del modelo a acciones que el robot entienda. 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.

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

Por mucho tiempo el hombre ha elaborado máquinas que imitan las partes del cuerpo humano. Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses, estos brazos fueron operados por sacerdotes, quienes clamaban que el movimiento de estos era inspiración de sus dioses. Los griegos contruyeron estatuas que operaban con sistemas hidráulicos para fascinar a los adoradores de los templos.

2.1. Robot

El inicio de la robótica actual puede fijarse en la industria textil del siglo XVIII, cuando Joseph Jacquard inventa en 1801 una máquina textil programable mediante tarjetas perforadas. La revolución industrial impulsó el desarrollo de estos agentes mecánicos, entre los cuales se destacaron el torno mecánico motorizado de Babbitt (1892) y el mecanismo programable para pintar con spray de Pollard y Roselund (1939). Además de esto durante los siglos XVII y XVIII en Europa fueron construidos muñecos mecánicos muy ingeniosos que tenían algunas características de robots. En 1805, Henri Maillardert construyó una muñeca mecánica que era capaz de hacer dibujos.

La palabra robot se empleó por primera vez en una obra de teatro llamada "Ros-sum's Universal Robota" (Los Robots Universales de Rossum) escrita por el dramaturgo checo Karel Čapek en 1920. La palabra checa 'Robota' significa servidumbre o trabajador forzado, y cuando se tradujo al inglés se convirtió en el término **robot**.

Sin embargo el término robot con el paso de los años ha cambiado, a continuación se listan algunas de las definiciones más aceptadas:

- La ISO 8373 lo define como “*un manipulador multipropósito automáticamente controlado y reprogramable de más de tres ejes*”.

- 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/*Instituto*.
- Mecanismo formado generalmente por elementos en serie, articulados entre sí, destinado al agarre y desplazamiento de objetos. Es multifuncional y puede ser gobernado directamente por un operador humano o dispositivo lógico[*Sanchez06*].

Para fines de estudio clasificaremos los robots en:

- Manipuladores. Generalmente están montados sobre una base fija que les sirve para definir su área de trabajo y su posición en la misma. Su estructura básica es la de un brazo humano.
- Móviles. Este tipo de robots tienen la capacidad de cambiar de posición por sí mismos para realizar alguna tarea determinada. Para lograrlo, tienen sensores que les permiten conocer su ambiente así como actuadores que le permiten desplazarse.

La gran mayoría de los robots usados en la industria son de tipo manipuladores.

2.1.1. Estructura de los robots manipuladores

Básicamente la estructura de un robot manipulador es la de un brazo articulado.

Se describe la estructura de un robot manipulador:

- Estructura mecánica.
- Sistema sensorial.
- Elementos terminales.
- Sistema de control.

Estructura mecánica

Desde el punto de vista mecánico, el robot está formado por una serie de elementos o eslabones unidos mediante articulaciones, que permiten un movimiento relativo entre cada dos eslabones consecutivos.

Existen diferentes tipos de articulaciones como podemos observar en la figura 2.1:

- 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 traslación a lo largo del eje de la articulación.
- c) En la articulación cilíndrica, existen dos grados de libertad, una rotación y una traslación 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.

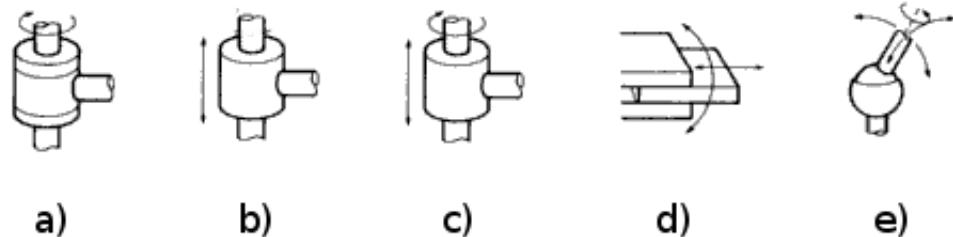


Figura 2.1: Tipos de articulaciones: a) rotación b) prismática c) cilíndrica d) planar e) esférica

Sistema sensorial.

Los sensores son transductores que convierten algún fenómeno físico en señales eléctricas que el microcontrolador puede leer [Jones98]. La principal función es trasladar la información del mundo real al mundo abstracto de la unidad lógica de procesamiento, para que ésta pueda responder de forma correcta.

Algunos de los tipos de sensores son:

- Proporcionan datos de movimiento
- Cambios iniciales
- Estructura externa del ambiente
- Los interruptores de contacto son dispositivos simples que reportan un valor binario: cerrado o abierto.

Elementos terminales.

Existen muchos tipos de motores, pero sólo unos cuantos son útiles en robótica, y dentro de éstos se encuentran los motores de corriente directa [Jones98]. Los motores o servomotores de CD son preferidos debido a su relativa facilidad para ser controlados, comparados con otros tipos de motores de CD y con los motores de CA [Estrada06].

Sistema de control.

El sistema de control en un robot es la parte más importante, ya que es aquí donde las acciones lógicas son convertidas en acciones físicas. Estas acciones son realizadas por actuadores y pueden depender de un evento del exterior captado por algún sensor.

2.1.2. Clasificación

Existe una gran variedad de tipos de brazos manipuladores y se pueden clasificar de distintas formas: por su estructura, por su forma de control, por su área de aplicación, por su fuente de potencia, por su geometría, por su movimiento cinemático, etc. [Spong89, Ramírez98]. Hoy en día, los manipuladores están agrupados en clases de acuerdo a la combinación de uniones usadas en su construcción [Petriu06] como puede observarse en la figura 2.2:

- Cilíndrico. Cuenta con dos articulaciones de rotación y una tipo planar.
- Esférico. Cuenta con una articulación de rotación y dos de tipo planar.
- S.C.A.R.A. Cuenta con dos articulaciones de rotación y una tipo planar.
- Cartesiano. Cuenta con tres articulaciones de tipo planar.
- Antropomórfico. Cuenta con tres articulaciones de tipo rotación.

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) son sistemas informáticos para realizar tareas de creación, modificación, análisis y optimización de un diseño.

Software CAM

El software CAM(Manufactura Asistida por Computadora, del inglés Computer Aided Manufacturing) son sistemas informáticos para la planificación, gestión y control de las operaciones de una planta de fabricación, mediante una interfaz directa o indirecta entre el sistema informático y los recursos de producción.

- Interfaz directa. Son aquellos donde la computadora se conecta directamente con el proceso de producción para monitorizar su actividad y realizar tareas de supervisión y control.

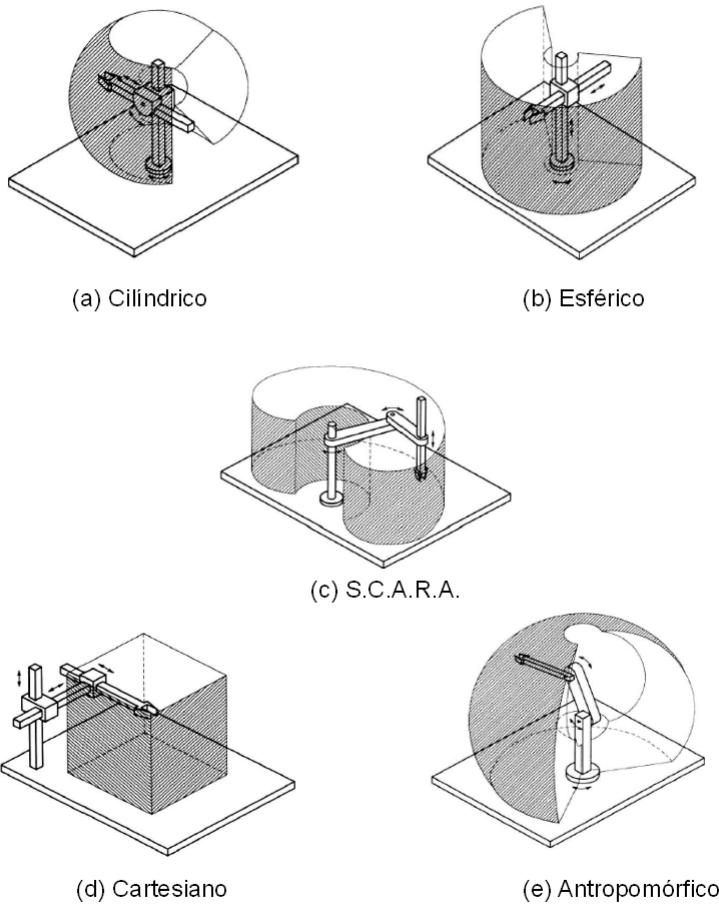


Figura 2.2: Categorías de los robots manipuladores y sus respectivas áreas de trabajo[Sanchez07].

- Interfaz indirecta. Se trata de aquellos en las que la computadora se utiliza como herramienta de ayuda para la fabricación, pero en las que no existe conexión directa con el proceso de producción.

La entrada de estos softwares son generalmente los diseños generados con software CAD.

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

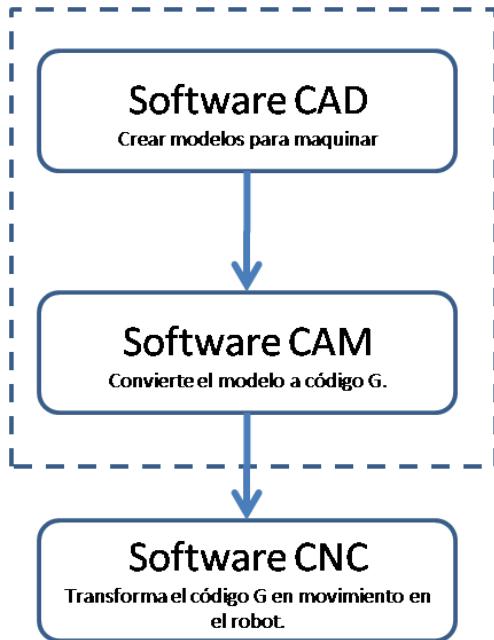


Figura 2.3: Programación de un robot manipulador.

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

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 sólo 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.

Algunos software que pueden ser aplicados a la programación de tareas son:

Software CAD

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

Software CAM

- ArtCAM
- DeskCNC
- MeshCAM

Software CNC

- TurboCNC
- EMC
- DeskCNC

2.2.2. Desarrollo tecnológico

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

Instituciones de nivel superior 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.

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.

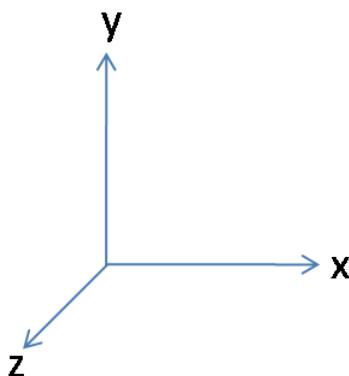


Figura 3.1: Sistema de coordenadas de 3 dimensiones

Sin embargo, la aplicación desarrollada utiliza sólo 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, es un elemento geométrico adimensional que describe una posición en el espacio. Un punto puede determinarse en el sistema de coordenadas cartesianas mediante las distancias a los ejes principales, que se indican con dos variables (x, y) en el plano y con tres variables (x, y, z) en el espacio tridimensional.

Para fines de programación se definió un punto en el espacio con el predicado $Punto(x, y, z)$.

3.2.2. Línea

Euclides, en su tratado denominado “Los elementos”[euclides09] ,establece varias definiciones para línea:

- Los extremos de una linea son puntos.
- Una linea es una longitud sin anchura.

La línea recta es la sucesión continua e indefinida de puntos en una sola dimensión[wikipedia09]. Su ecuación general es:

$$y = mx + B \quad (3.1)$$

$$m = \frac{(y_n - y_0)}{(x_n - x_0)} \quad (3.2)$$

donde m es la pendiente y y es la intercepción con el eje Y. En la figura 3.2 se puede observar un ejemplo de ésta.

Para fines de programación se definió con el predicado $Línea(a, b)$, donde a y b son dos puntos diferentes cualesquiera en el espacio.

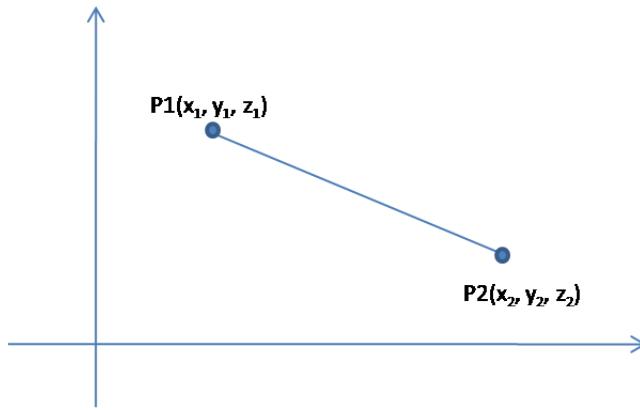


Figura 3.2: Línea

3.2.3. Círculo

Es una secuencia de puntos infinitos del plano equidistantes de otro fijo, llamado centro; la distancia del centro a cualquier punto se denomina radio(r). Su ecuación paramétrica puede apreciarse en la ecuación 3.3. En la figura 3.3 se puede observar un ejemplo.

$$\begin{aligned} x &= r * \cos(\Theta) \\ x &= r * \sin(\Theta) \\ \Theta &\in [0, 2\pi] \end{aligned} \tag{3.3}$$

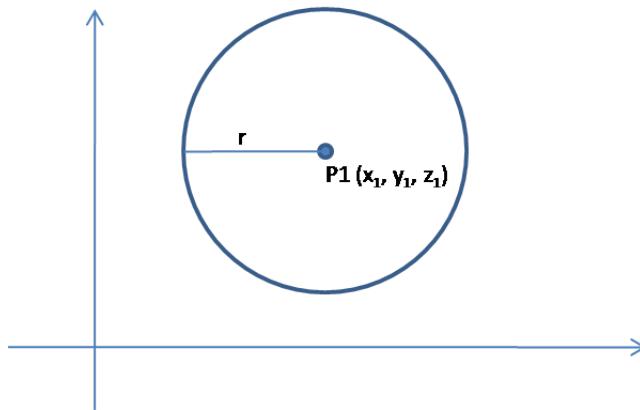


Figura 3.3: Círculo

Para fines de programación se definió con el predicado *Circulo*(c, r) donde c es el punto definido como centro y r es el radio o distancia del centro a cualquier otro punto.

3.2.4. Elipse

Es una secuencia de puntos infinitos del plano tales que la suma de la distancia a dos puntos fijos llamados focos es una constante positiva e igual a la distancia entre los vértices. Su ecuación paramétrica puede apreciarse en la ecuación 3.4. En la figura 3.4 se puede observar un ejemplo.

$$\begin{aligned}
 x &= r_1 * \cos(\Theta) \\
 y &= r_2 * \sin(\Theta) \\
 \Theta &\in [0, 2\pi]
 \end{aligned} \tag{3.4}$$

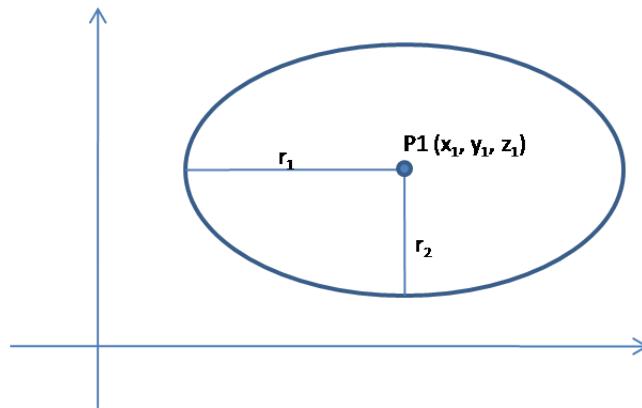


Figura 3.4: Elipse

Para fines de programación se definió con el predicado *Elipse* (c , r_1 , r_2) donde c es el punto definido como centro, r_1 es la distancia al punto más alejado en el eje horizontal y r_2 es la distancia al punto más alejado sobre el eje vertical.

3.2.5. Curva de Bezier

3.2.6. Elementos compuestos

Es un objeto compuesto por una o más primitivas para formar un elemento compuesto. Este elemento puede contener infinitas primitivas dentro de si misma. En la figura 3.5 se muestra un ejemplo de un elemento compuesto, este elemento contiene tres líneas.

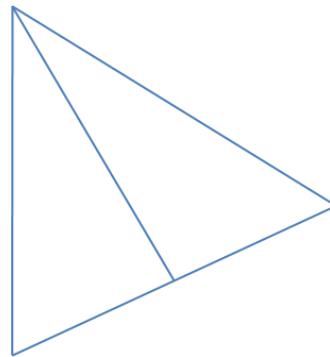


Figura 3.5: Elemento compuesto

3.3. Transformaciones

Como se mostró en las secciones anteriores, las primitivas se están definiendo mediante puntos de control, un punto en el espacio 3D homogéneo será un vector columna como el que se muestra en la ec. 3.5.

$$p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.5)$$

3.3.1. Escalado

A continuación se muestra como quedaría el escalado de un punto de control.

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & e_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.6)$$

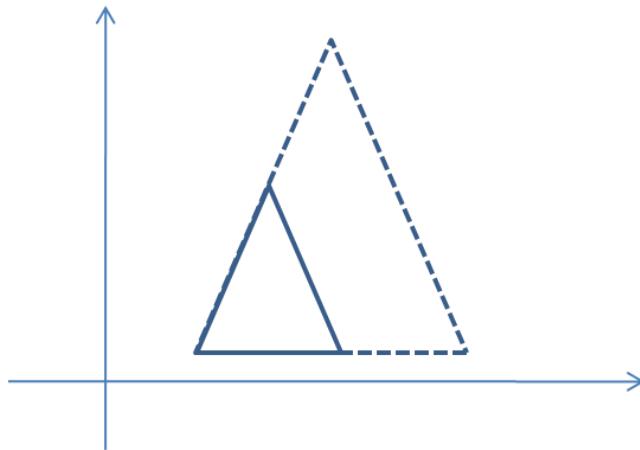


Figura 3.6: Escalamiento

3.3.2. Traslación

La traslación $x' = x + t_x, y' = y + t_y, z' = z + t_z$ puede ser expresada como el producto:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.7)$$

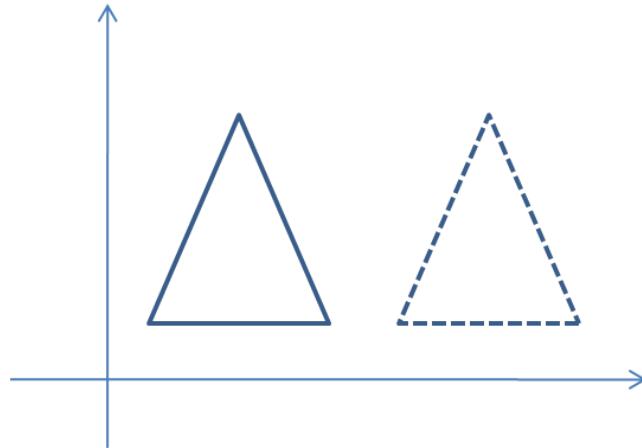


Figura 3.7: Traslación

3.3.3. Rotación

La rotación un ángulo Θ alrededor del eje z es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & -\sin(\Theta) & 0 \\ 0 & \sin(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.8)$$

La rotación un ángulo Θ alrededor del eje y es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & 0 & -\sin(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.9)$$

La rotación un ángulo Θ alrededor del eje x es:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) & 0 & 0 \\ \sin(\Theta) & \cos(\Theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3.10)$$

3.4. Almacenamiento y recuperación

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

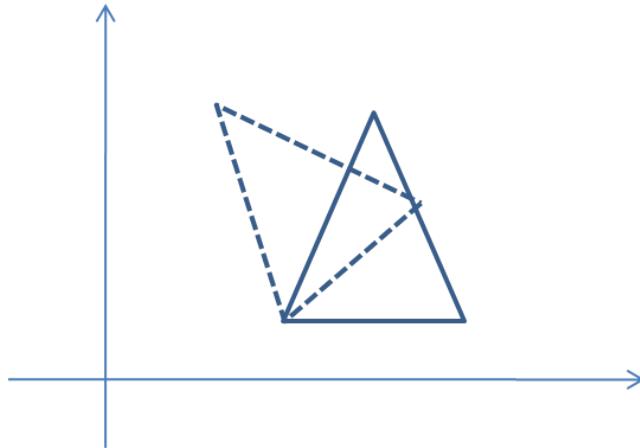


Figura 3.8: Rotación

3.4.1. 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.

3.4.2. Estructura de almacenamiento

La estructura de almacenamiento puede observarse en la figura 3.9. Como puede verse la estructura es simple, compacta y fácil de entender.

En la figura 3.9 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 un número entero (tipo):
 - Punto (1)
 - Línea (2)
 - Poli-Línea (3)
 - Círculo (4)

```

<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 3.9: Estructura de almacenamiento

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

3.5. Importación de otros formatos

3.5.1. Imágenes PGM

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

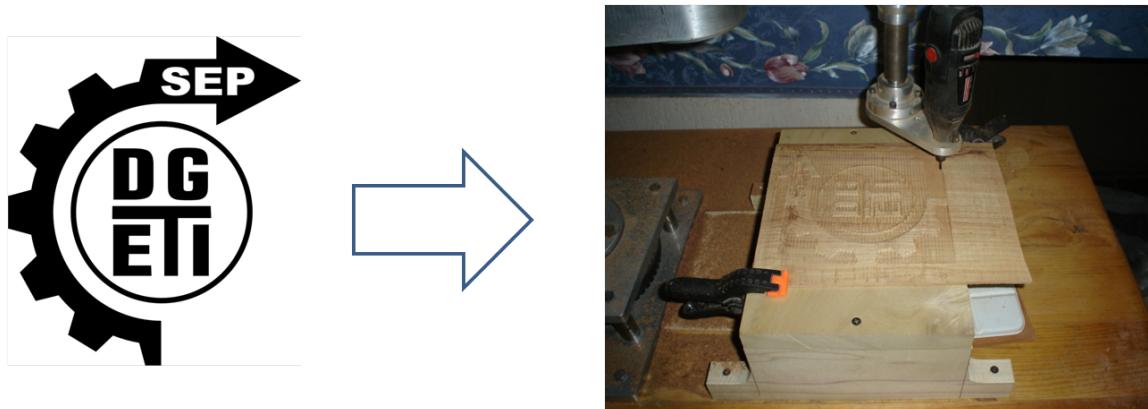


Figura 3.10: PGM a Robot

Capítulo 4

Interfaz de software

La interfaz de usuario es uno de los apartados con más relevancia de un sistema de cómputo, en la actualidad todo sistema debe contener una herramienta con la que el usuario pueda ordenar a la computadora qué 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:

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

4.1. Diseño de la interfaz de usuario

La interfaz gráfica de usuario (IGU) es la parte más importante de los sistemas computarizados debido a que es con la que el usuario interactúa de forma directa. Las metas de una interfaz de usuario son: hacer que el trabajo con la computadora sea fácil, productivo y agradable[Galitz07].

La interfaz de usuario se conforma de dos componentes: entrada y salida. La entrada es la forma en que los usuarios comunican sus necesidades a la computadora. La salida es el medio por el cual la computadora muestra los resultados de las operaciones requeridas por el usuario.

Propiamente, una interfaz de usuario provee la mezcla de: mecanismo de entrada y salida que de manera eficiente satisfagan las necesidades del usuario, capabilities, y limitaciones en la forma más eficiente.

4.1.1. Principios generales de un buen diseño

Estos principios son características generales para la interfaz en todos sus aspectos:

Accesibilidad

El sistema deberá de ser usado por usuarios de diversas capacidades sin diseños especiales ni modificaciones. Originalmente este término fue usado para referirse a sistemas accesibles a usuarios con discapacidades. Actualmente el término accesibilidad se refiere a cubrir la mayoría de los usuarios.

Las principales características de un sistema accesible son:

- Perceptibilidad
- Operatividad
- Simplicidad

Disponibilidad

Todos los aspectos de un sistema deberán de estar disponibles en cualquier momento en cualquier tarea. Solo deberán de no estar disponibles en aquellas tareas que así se definan.

Claridad

La interfaz debe ser clara en apariencia visual y concepto. Los elementos visuales deben ser entendibles, relacionados con objetos del mundo real. Los conceptos y textos deben ser simples y no confusos.

Consistencia

Consistency is uniformity in appearance, placement, and behavior within the user interface.

Es importante ya que puede reducir la necesidad de adquirir nuevas habilidades para una actividad que puede ser similar a otra. Si un nuevo sistema impone necesidades de aprender nuevas habilidades en los usuarios, este puede convertirse en un sistema no productivo e innecesario.

Un sistema deberá lucir y operar de forma normal(throughout), es decir:

- Una función siempre deberá devolver el mismo resultado.
- La posición de los elementos estándares no cambiará.

Control

El usuario debe tener control sobre las acciones que está realizando el sistema.

Eficiencia

En cada paso de un proceso, se debe mostrar al usuario toda la información y herramientas necesarias para terminar el proceso. El usuario no debe tener la necesidad de buscar información ni herramientas en lugares externos al proceso activo.

Tiempo de respuesta

Cuando un usuario realice una solicitud el sistema debe contestar lo más rápido que le sea posible. Si al realizar una solicitud el sistema no responde en un tiempo considerable, puede ser considerado como que el sistema ha fallado.

Recuperacion

El sistema debe contar con un sistema de recuperación que permita deshacer una acción. Con este mecanismo se reduce en gran medida los errores de usuarios nuevos. El punto de retorno puede ser hacia la acción anterior, pantalla anterior o al inicio de un determinado periodo de tiempo.

El ojetivo es mantener la estabilidad, es decir, cuando el usuario realice una acción errónea que pueda llevar a una situación peligrosa exista alguna forma de regresar al puto anterior estable. La recuperación debe ser obvia, automática, y simple de realizar.

Simplicidad

La simplicidad es reconocida cuando cualquier usuario puede de manera simple entender y usar un sistema con la mínima experiencia y documentación.

Visibilidad

Los sistemas son más usables cuando de una manera clara indican el estado y los resultados de las acciones realizadas por los usuarios.

4.2. 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.

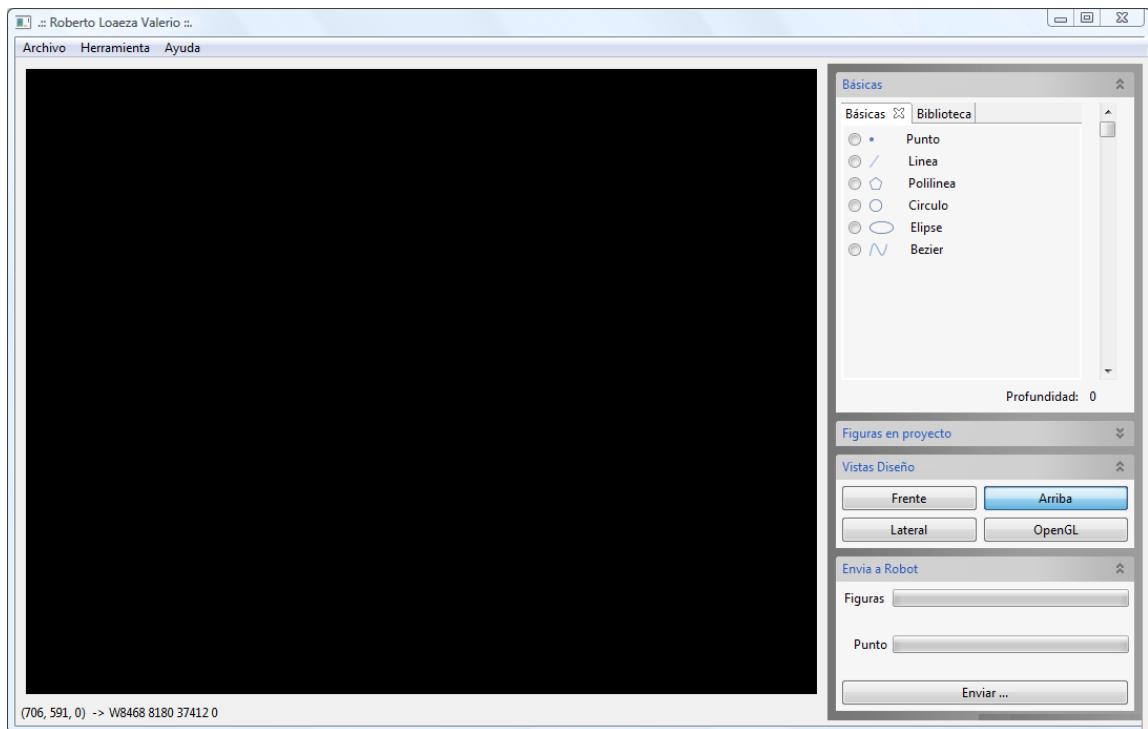


Figura 4.1: Diseño de la interfaz gráfica

- .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.

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.

	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

4.2.1. 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.

4.3. 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 deshechada 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 una 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 extensiones 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 ésta llega a colapsar o mal-funcionar, 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 la Facultad de Ingeniería Electrica de la Universidad Michoacan de San Nicolas de Hidalgo, fijo en su base y hecho de acero y aluminio.

La estructura física fue realizada por una empresa dedicada a la robótica, denominada “Cervantes Co.”, ubicada en Paracho Michoacán.

La programación del controlador fue realizada por Omar Rodríguez Páez[Rodriguez08] .

5.1.1. Características

Entre las características más relevantes acerca del robot se tiene:

- 4 Grados de libertad.
- Altura 60 cm.
- Alcance máximo de su brazo, 50 cm.
- Construido en acero y aluminio.
- Peso alrededor de 45 Kg.

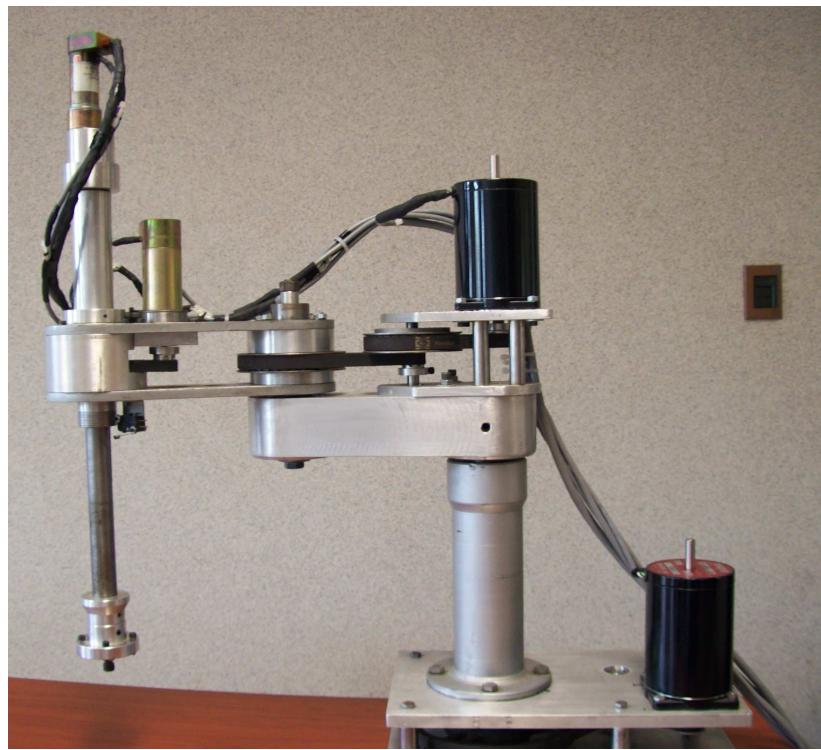


Figura 5.1: Robot SCARA

5.1.2. Arquitectura

Como se mencionó en la sección++ la arquitectura de un robot es:

Estructura mecánica.

Sistema sensorial.

El robot cuenta con sensores:

- Sensores Odométricos. Estos sensores mide la posición o rango de rotación de un eje.
- Sensores de límite o de posición límite de cada articulación. Es un dispositivo muy simple que provee una gran fiabilidad para conocer con certeza las posiciones de inicio y final de cada uno de los ejes del robot SCARA.

Además los convertidores analógico-digitales del microcontrolador, se usan como sensores de voltaje para medir indirectamente la corriente que circula por los motores a través de los controladores.

Elementos terminales.

El robot cuenta con dos motores de pasos, uno para el hombro y otro para el codo, éstos permiten un movimiento sobre los ejes X y Y; y dos motores de CD con escobillas para el eje Theta y Theta-Z.

Sistema de control.

El robot utiliza un sólo módulo Adapt9S12DP256 con el microcontrolador 9S12DP256C de Motorola [Arts04], el cual se muestra en la figura 5.2.

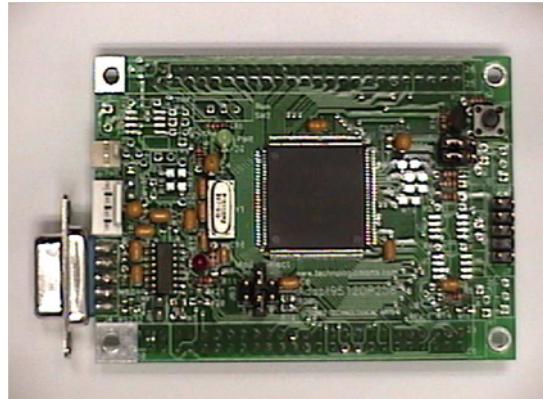


Figura 5.2: Módulo Adapt9S12DP256 del microcontrolador 68hcs12[Rodriguez08]

El lenguaje soportado se lista en el apendice A.

5.1.3. Limitantes

5.2. Conversión de modelado a acciones SCARA

Debido a que en el modelo sólo contamos con puntos definidos dentro del sistema de coordenadas cartesianas de tres dimensiones es necesario calcular el conjunto de ángulos de las articulaciones para logar 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 ángulos 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 la solución de forma cerrada, en algunos casos las soluciones numéricas son tan lentas que pueden ocasionar problemas de cinemática, es por esto que sólo nos enfocaremos en las soluciones analíticas, en particular en la solución geométrica.

Solución Geométrica

Recordando algunas identidades trigonométricas:

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

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

,

$$\cos(\Theta) = \cos(-\Theta) \quad (5.3)$$

Primero se sabe que la orientación del último eslabón es la suma de las variables articulares, como se muestra en la figura 5.3.

$$\Theta = \Theta_1 + \Theta_2 + \Theta_3 \quad (5.4)$$

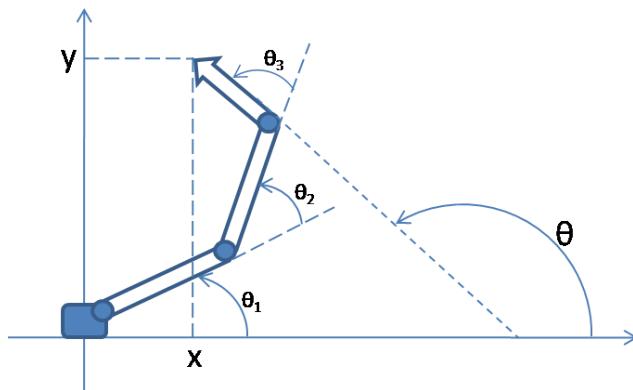


Figura 5.3: Orientación del último eslabón

Se calcula θ_2 aplicando ec. 5.2:

$$x^2 + y^2 = l_1^2 + l_2^2 - (2l_1l_2) \cos(180 - \Theta_2) \quad (5.5)$$

debido a que:

$$\cos(180 - \Theta_2) = -\cos(\Theta_2) \quad (5.6)$$

Resulta:

$$\cos(\Theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \quad (5.7)$$

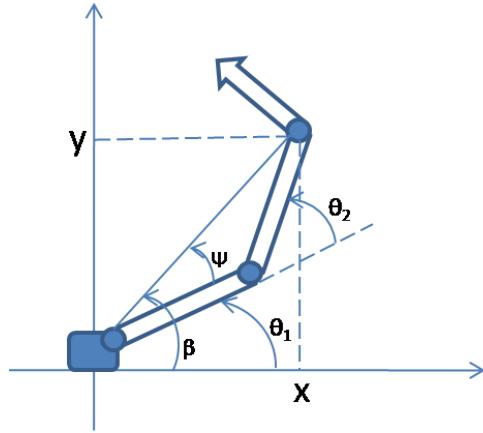


Figura 5.4: Ángulos β , ψ y Θ_1

Se calcula θ_1 :

Si se definen dos ángulos como se muestra en la figura 5.4 se cumple:

$$\Theta_1 = \beta - \psi \quad (5.8)$$

El ángulo β se calcula:

$$\sin(\beta) = \frac{y}{\sqrt{x^2 + y^2}} \quad (5.9)$$

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

$$\cos(\psi) = \frac{x^2 + y^2 + l_1^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}} \quad (5.10)$$

Finalmente se calcula θ_3 usando la siguiente ecuación:

$$\Theta_3 = \Theta - \Theta_1 - \Theta_2 \quad (5.11)$$

5.3. Conectividad PC-Robot

Debido a que el robot SCARA usa conexión mediante el puerto serial (RS232) primero se realizó una interfaz capaz de comunicar la aplicación desarrollada en lenguaje Java, con el sistema de control del robot SCARA.

Existen varias bibliotecas para la comunicación serial compatibles con lenguaje Java, algunas bibliotecas aún están en desarrollo, otras sólo son para un determinado sistema operativo. En la tabla 5.1 se muestran algunas de las bibliotecas más usadas para la conexión mediante puerto serial.

Biblioteca	Multiplataforma	Tipo de licencia
javaconn	No	GPL
rxtx	Si	GPL

Tabla 5.1: Bibliotecas para comuniación RS232

Se utilizó la biblioteca RXTX para realizar la conexión con el 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

Comandos soportados por el robot SCARA

A continuación se listan los comandos soportados por el robot SCARA

Comando para mover los motores del robot

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/>
- [Kurfess05] Kurfess, T. Robotics and Automation Handbook. Ed. CRC PRESS, 2005.
- [Gibilisco03] Gibilisco, S. Concise Encyclopedia of Robotics, Ed. McGraw-Hill, 2003
- [Davison07] Davison, A. Pro Java 6 3D Game Development Java 3D, JOGL, JInput and JOAL APIs. Ed. Apress, 2007
- [Ollero] Ollero, A. Robótica: Manipuladores y robots móviles, Marcombo Editorial.
- [Galitz07] Galitz, W. The Essential Guide to User Interface Design, Ed. Wiley, 2007.
- [Rodriguez08] Rodriguez, O. Diseño y Construcción de un Robot SCARA utilizando motores de CD de pasos y con escobillas, Tesis de Licenciatura, UMSNH, 2008
- [Estrada06] Estrada, C. Diseño de un sistema de control de seguimiento de trayectorias para un robot móvil. Tesis de Maestría. UMSNH, 2006.
- [Sanchez07] Sánchez, O. Cinemática de los manipuladores, Universidad Huelva, Huelva España. 2007
- [Sanchez06] Sánchez, O. Robot, Universidad Huelva, Huelva España. 2006
- [Instituto] Instituto de Robótica de América
- [Kurfess05] Kurfess, T .Robotics and automation handbook. Ed. CRC Press, 2005
- [Delrieux00] Delrieux, C. Introducción a la Computación Gráfica. Dpto Ingeniería Eléctrica, Universidad nacional del sur. 2000.
- [euclides09] www.euclides.org: Los elementos
- [wikipedia09] www.wikipedia.org: Recta
- [Jones98] Jones, J. L., Flynn, A. M., y Seiger, B. A. Mobile Robots. Cambridge University Press, 1998.