Capítulo 5 Implementación de ROBIN

Este capítulo muestra el sistema implementado, la información se dividió en los cuatro módulos principales que conforman el sistema ROBIN: el módulo de construcción, módulo de control, módulo de programación y módulo de simulación. Por cada uno de los módulos se explican los problemas que se tuvieron que resolver y de que manera se resolvieron.

El sistema se desarrolló en el lenguaje de programación C++. Se utilizó la versión de Microsoft Developer Strudio 97 con Visual C++ 5.0 como ambiente de desarrollo en una máquina PC Pentium II (350 Mhz) con 32 Mb en RAM con sistema operativo Windows 98. Las gráficas se desarrollaron bajo el estándar gráfico Open GL y en la interfaz se utilizaron las librerías de clases de Microsoft (comúnmente llamadas MFC) para Windows.

El nombre del sistema es ROBIN por **rob**ot **in**dustrial y se describe como una herramienta gráfica de simulación de robots industriales en un ambiente virtual.

5.1 Módulo Construcción

Esté módulo se encarga de construir y visualizar el mundo virtual como se menciono en la sección 4.1.1. Para este módulo se utilizó una librería gráfica ya existente basada en Open GL. Dentro de las principales ventajas de esta librería se encontraba que su eficiencia estaba ya probada y que las figuras y funciones que ofrece están muy completas. El nombre de esta librería es GLOOP (OpenGL Object Oriented Programming library). Esta librería fue realizada por Craig Fahrnbach [97] y utiliza un formato en texto para almacenar las figuras, lo que la hacen fácil de usar y programar. GLOOP implementa una serie de primitivas básicas y sus operaciones de traslación, rotación, color y hasta textura. Además de presentar una muy buena documentación, su eficiencia y calidad la convirtieron en la mejor opción para el sistema.

Para construir el mundo tenemos dos opciones, la primera es dentro del sistema seleccionamos del menú **Archivo** la opción de **Nuevo** o de la barra de tareas el ícono  después iremos agregando los objetos que necesitamos. La otra opción es leyendo un archivo ya creado que contenga la descripción de este mundo, estos archivos contienen la

extensión wld (por mundo). Para agregar objetos se utiliza ya sea la barra de menus (Objeto-Figura) o la barra de icónos (ver Figura 5.1). Los objetos con que contamos son: cono, cilindro, cubo, sólido de revolución, esfera, toroide, texto, disco, dxf, plano, luz y robot.

Figura 5. 1 Agregando un objeto

Al momento de crear un objeto, aparecerá un cuadro de diálogo que nos permitirá modificar sus distintos atributos. La Figura 5.2 muestra el cuadro de diálogo de un cilindro. Estos diálogos nos permiten modificar los diferentes atributos de la figura correspondiente

Figura 5. 2 Atributos del objeto cilindro.

La definición varía de acuerdo al tipo de figura pero los demás diálogos: Origen, Rotación, Escala, Traslación, Color/Material, Textura y Animación son comunes a todos. Los diálogos de Origen, Rotación, Traslación y Escala corresponden al módulo de control, este módulo se describe en la sección 5.2.

El diálogo de Color/Material nos permite asignarle el color que deseamos al objeto. El color lo podemos seleccionar de la paleta de colores. Esta paleta de colores tiene la forma del cubo RGB. Para seleccionar el color colocamos el cursor sobre él y presionamos el botón izquierdo del ratón. También podemos escribir sus valores RGBA. La Figura 5.3 muestra este diálogo. Si queremos agregar textura en el diálogo tendremos que agregar el nombre del archivo de la textura correspondiente.

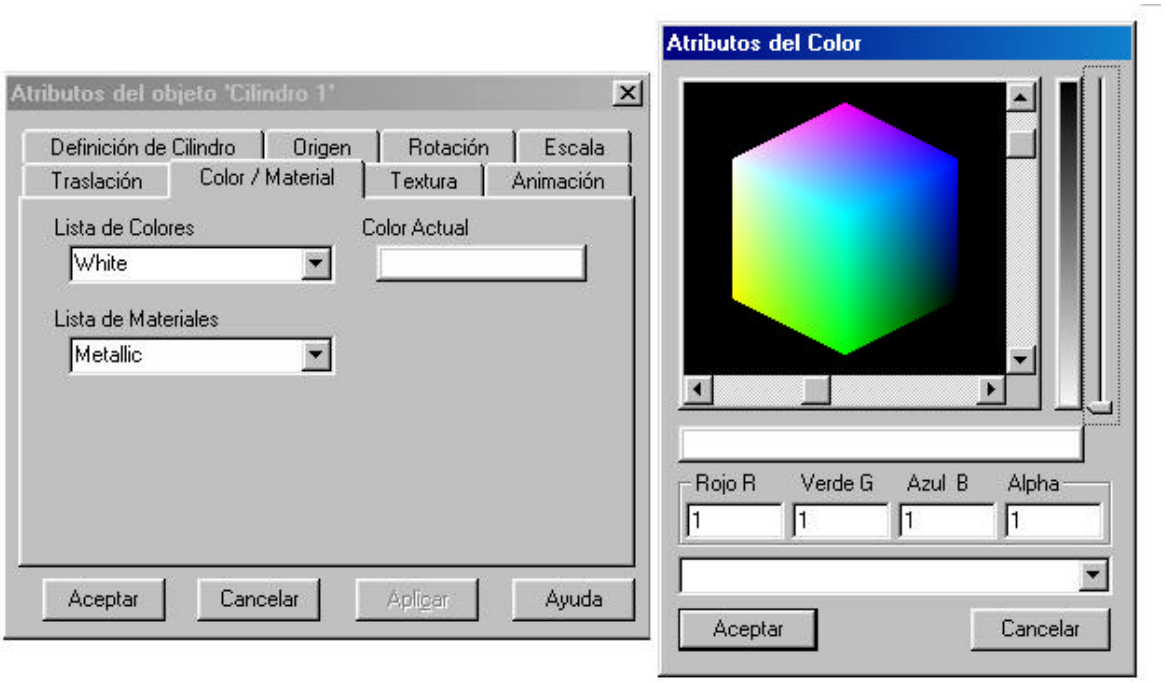


Figura 5. 3 Diálogo de Color/Material

Una vez especificados sus atributos, nombre, tamaño, color, posición y orientación, podremos ver en la pantalla el objeto. Si se desea redefinir estos atributos lo único que tenemos que hacer es seleccionar el objeto ya sea en la ventana principal o en la vista de árbol y hacer doble clic.

La clase principal es la clase mundo y al agregar cada objeto lo que hacemos es añadir este a su lista de objetos para que al momento de redibujarse la pantalla lo podamos ver. Cuando necesitamos crear estructuras más complejas con más de una figura, lo que tenemos que hacer es seleccionar al objeto padre antes de agregar el nuevo objeto. De esta manera el objeto que creemos formará parte del objeto padre. Esto significa que si movemos o rotamos el objeto padre, sus objetos hijos lo seguirán. Este tipo de figuras son muy útiles para crear estructuras complejas, la Figura 5.4 muestra como ser construye una mesa.

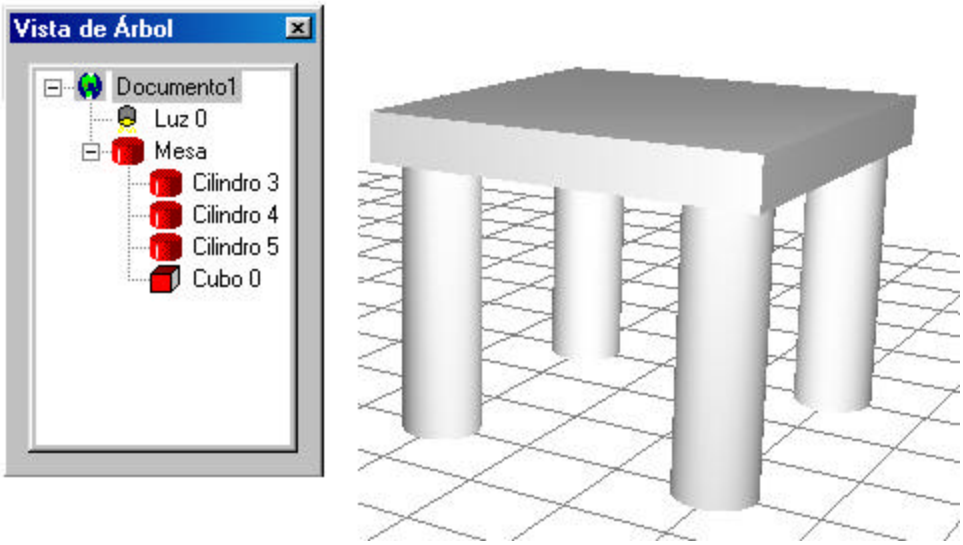


Figura 5. 4 Mesa

Una vez que creamos un objeto es posible guardarlo, esto es muy útil para cuando vamos a utilizar el objeto muchas veces y no queremos estarlo "construyendo" cada vez. Para almacenar un objeto seleccionamos del menú **Objeto** la opción de **Guardar como ...** Este tipo de archivos tienen la extensión *obj* y consiste en una lista de todos los atributos del objeto. Cuando queremos agregar al mundo un objeto almacenado seleccionamos del menú **Objeto** la opción de **Abrir**.

Para guardar el mundo que creamos, seleccionamos del menú **Archivo** la opción de **Guardar** o de la barra de tareas el ícono  . La extensión del archivo del mundo es *wld*. Cuando queremos recuperar un mundo almacenado seleccionamos del menú **Archivo** la opción de **Abrir**. o de la barra de tareas el ícono  . Estas operaciones se llevan a cabo por la función *Serialize* nativa de todos las clases en Windows. Esta función recibe un

archivo y de acuerdo a la operación almacena o lee. La manera en que la clase mundo se almacena es la siguiente: primero se escriben en el archivo las propiedades de la cámara, después los atributos del mundo y por último los atributos de todos los objetos Al momento de leer un archivo, se construye el mundo y la cámara y se leen sus atributos. Después se va creando cada objeto descrito en el archivo y se agrega al mundo. A continuación se muestra un fragmento de un archivo *wld*.

```
// Version: 1.40
// World Definition File
// 'glOOP' version 1.40,
// All rights reserved
// OpenGL is a registered trademark of Silicon Graphics

#include   < Default.mtl >  //Our Worlds Materials

C3dCamera {
 Origin    < 0.000000 2.000000 15.000000 >
 Rotation  < -75.000000 0.000000 -15.000000 >
 Perspective View  < 1 >
 Field of View 'Y' < 45.000000 >
 Near clip Plane   < 1.000000 >
 Far clip Plane    < 5000.000000 >
 CAnimation Procedures {
 }
}

C3dWorld {
 Name         < glDocument1.wld >
 DisplayAxis  < 1 >
 DisplayGrid  < 1 >
 DisplayMode  < 2 >
 AnimateWorld < 0 >
 Enable Fog   < 0 >
...
C3dObjectCone {
 Solid         < 1 >
 Height        < 2.000000 >
 BaseRadius    < 1.000000 >
 Smooth        < 1 >
 Segments      < 40 >
 Solid Color   < 1 >
...
 Name          < Cone 0 >
 Color         < 0.678431 0.211765 1.000000 1.000000 > // RGBA
 Origin        < -0.228666 0.059807 0.436473 0.000000 >
 Rotation      < 0.000000 0.000000 0.000000 >
 Scale         < 1.000000 1.000000 1.000000 >
 Translate     < 0.000000 0.000000 0.000000 >
 Locks         < 0 0 0 0 0 0 >
```

Tomando de base librería GLOOP y las ventajas que ofrece la programación orientada a objetos se implementaron dos tipos de clases indispensables para este proyecto. La primera está encargada de leer archivos DXF. DXF es el formato utilizado por Autocad

de Autodesk y es un estándar en la industria. El objetivo de programar esta clase es el de ofrecer figuras más apegadas a la realidad ya que los modelos DXF están formados por conjuntos de triángulos y polígonos que conforman una figura tridimensional. En la Internet es posible encontrar una gran variedad de modelos y figuras en este formato. La implementación de esta clase se basó en un programa de Ian L. Kaplan [90]. Su programa analiza un archivo DXF y obtiene los datos referentes a las caras del objeto tridimensional. Este se muestra en forma de líneas solamente, la clase implementada permite visualizar los archivos DXF con caras con color y textura, además de incorpora las funciones que ofrece la librería GLOOP para sus primitivas. La Figura 5.5 presenta el modelo DXF de la carrocería de un automóvil.

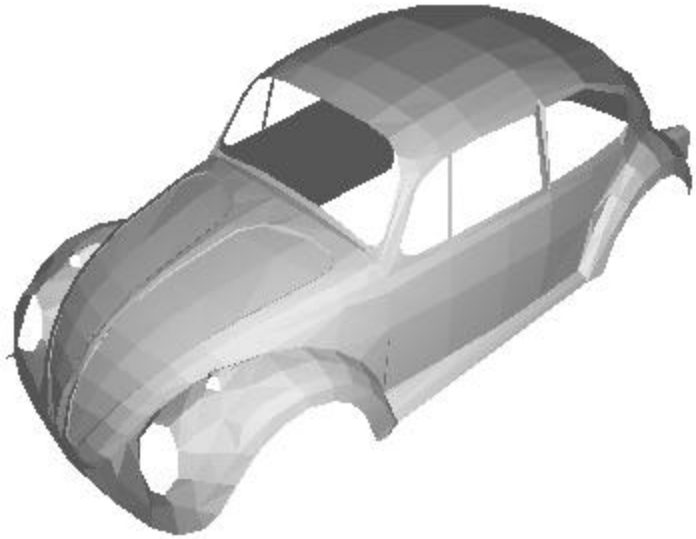


Figura 5. 5 Carrocería de automóvil

La segunda clase que se implementó fue la clase Robot, esta clase es la encargada de contener todas las piezas que conforman al robot además de controlarlas. Esta clase requiere de un archivo de texto con la notación Denavit-Hartenberg para su control. Y en el caso de contar con un programa para el robot se incluye este para que lo ejecute posteriormente. Un programa consiste en una serie de puntos por los que pasará el elemento terminal del robot. En el módulo de programación se revisará la estructura de un programa para el robot (ver sección 5.3).

La notación Denavit-Hartenberg (DH) planteada por primera vez en 1955 por Denavit y Hartenberg, de ahí su nombre, es una notación que permite representar cualquier mecanismo (no solamente robots) en una matriz de cuatro parámetros por cada eslabón de una cadena articulada. Según la representación DH, escogiendo adecuadamente los sistemas de coordenadas asociados a cada eslabón será posible pasar de uno al siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón. Este método permite reducir el problema cinemático directo a encontrar una matriz de transformación homogénea 4x4 que relaciona la ubicación del extremo del robot con respecto al sistema de coordenadas de su base.

Los cuatro parámetros de DH que dependen de las características geométricas de cada eslabón y las articulaciones que le unen con el anterior y el siguiente representan [Barrientos *et al.*];

θ_i Es el ángulo que forman los ejes $\mathbf{x}_{i-1}$ y $\mathbf{x}_i$ medido en un plano perpendicular al eje $\mathbf{z}_{i-1}$ utilizando la regla de la mano derecha. Se trata de un parámetro variable en articulaciones giratorias.

d_i Es la distancia a lo largo del eje $\mathbf{z}_{i-1}$ desde el origen del sistema de coordenadas (i-1)-ésimo hasta la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$. Se trata de un parámetro variable en articulaciones prismáticas.

a_i Es la distancia a lo largo del eje $\mathbf{x}_i$ que va desde la intersección del eje $\mathbf{z}_{i-1}$ con el eje $\mathbf{x}_i$ hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En el caso de articulaciones prismáticas, se calcula como la distancia más corta entre los ejes $\mathbf{z}_{i-1}$ y $\mathbf{z}_i$.

α_i Es el ángulo de separación del eje $\mathbf{z}_{i-1}$ y el eje $\mathbf{z}_i$, medido en un plano perpendicular al eje xi, utilizando la regla de la mano derecha.

La Figura 5.2 muestra un ejemplo a partir de un eslabón giratorio. En 1986 John J. Craig [86] propuso un formato de los parámetros de DH el cual se muestra en la Tabla 5.1.

i	α_{i-1}	a_{i-1}	d_i	θ_i

Tabla 5. 1 Formato de notación DH según John Craig.

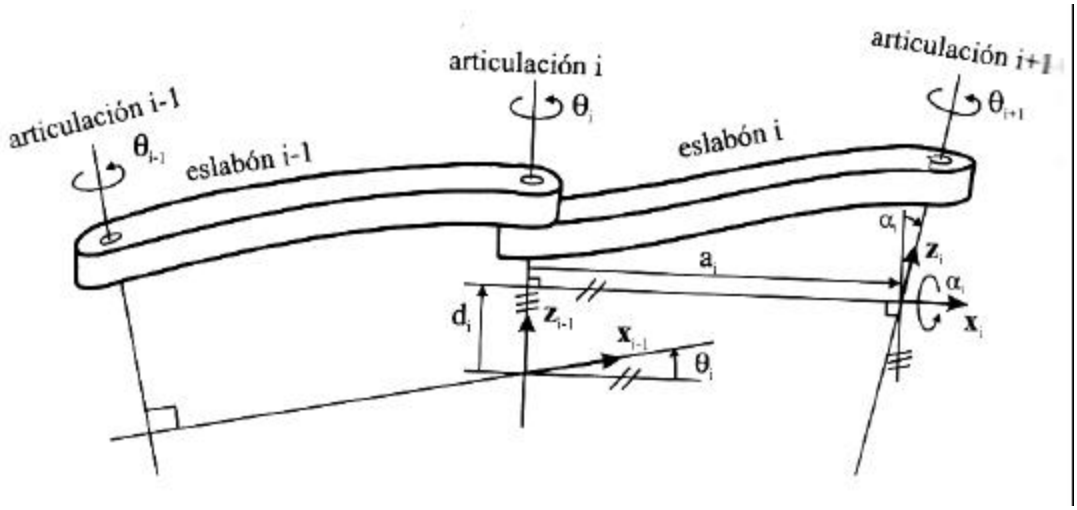
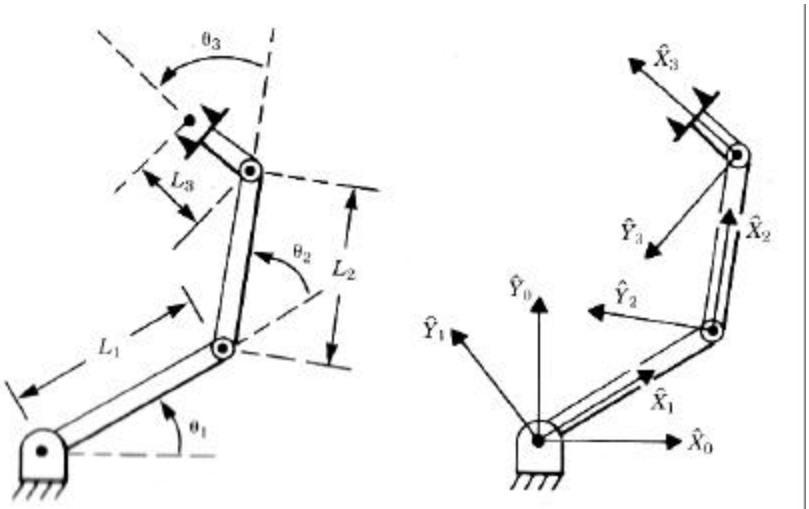


Figura 5. 6 Parámetros D-H para un eslabón giratorio

En la Figura 5.3 se puede ver la manera en que este formato se compone a partir de la descripción geométrica de los eslabones de un robot articulado RRR. Este ejemplo se tomo del libro de John Craig.

Figura 5. 7 Notación DH para un robot RRR.



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3

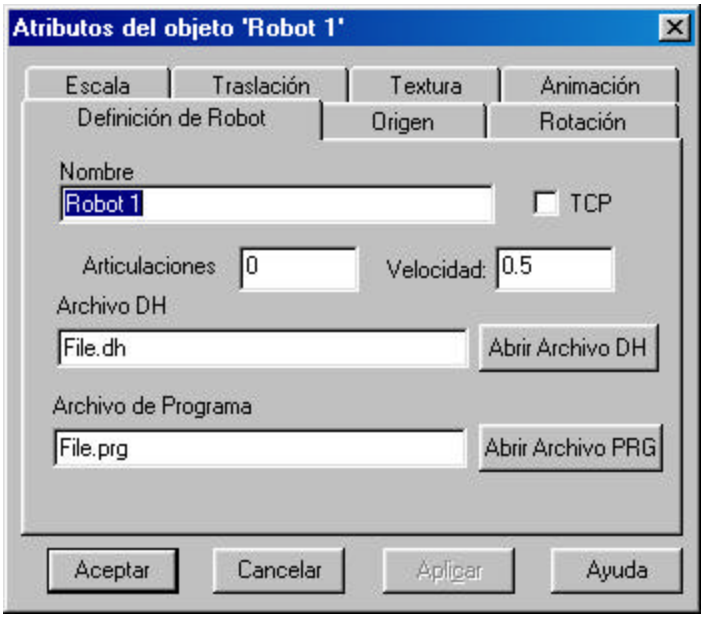
De esta manera podemos representar cualquier robot, cabe señalar que la configuración que obtenemos a partir de un robot no es única, pero debemos escoger la que más nos convenga de manera que podamos definir todo el robot. A continuación se muestra el archivo DH para un robot RRR como el de la Figura 5.7.

```
# This is a DH parameter file for the 3 DOF Planar Arm
# The variable is specified by typing a "var' or a "VAR" in its
# field. i.e. If the joint is revolute then it will be in the fourth
# coulumn and if the joint is prismatic it will be in the third
column
# The notation is based on the "Introduction to Robotics, Mechanics
and Controls"
# book by John Craig.

#    Alpha(i-1)    a(i-1)          d(i)            theta(i)

     0.0           0.0           0.0           var    # link 0-1
     0.0           25.0          0.0           var    # link 1-2
     0.0           25.0          0.0           var    # link 2-3
```

Como se puede ver el primer renglón corresponde a la primera articulación que es de rotación. El segundo renglón corresponde a la siguiente articulación de rotación y que se encuentra sobre el mismo plano que la anterior solo que el valor de 25.0 representa un desplazamiento de 25 unidades. Por último la tercera articulación de rotación también se encuentra a 25 unidades a partir de la segunda articulación

Creación de un robot. Para crear un robot se elige del menú **Objeto** la opción **Robot** o de la barra de figuras el ícono  . A continuación nos aparecerá el cuadro de diálogo para su definición (ver Figura 5.8).

Figura 5. 8 Creación de un robot.



Una vez creado el robot, se agregan las figuras que lo forman. La construcción del robot debe ser congruente con su descripción en el archivo DH. Para crear una articulación prismática o de rotación, es necesario hacer tres cosas:

1. Escoger la figura (cilindro, cubo) que va a representar a la articulación y agregarla al objeto robot.

2. Nombrar a la figura como "articulación *n*" donde *n* es el número de articulación.

3. Después de colocar la figura en la posición deseada, bloquear el movimiento y rotación de la articulación en todos los ejes de coordenadas. Si la articulación es de rotación se deja desbloqueado el eje Z para rotación. En cambio si la articulación es prismática se desbloquea el eje Z para movimiento. En caso de existir restricciones tanto de giro como de desplazamiento, señalarlo en los atributos del objeto

La Figura 5.9 muestra el robot RRR en el sistema:

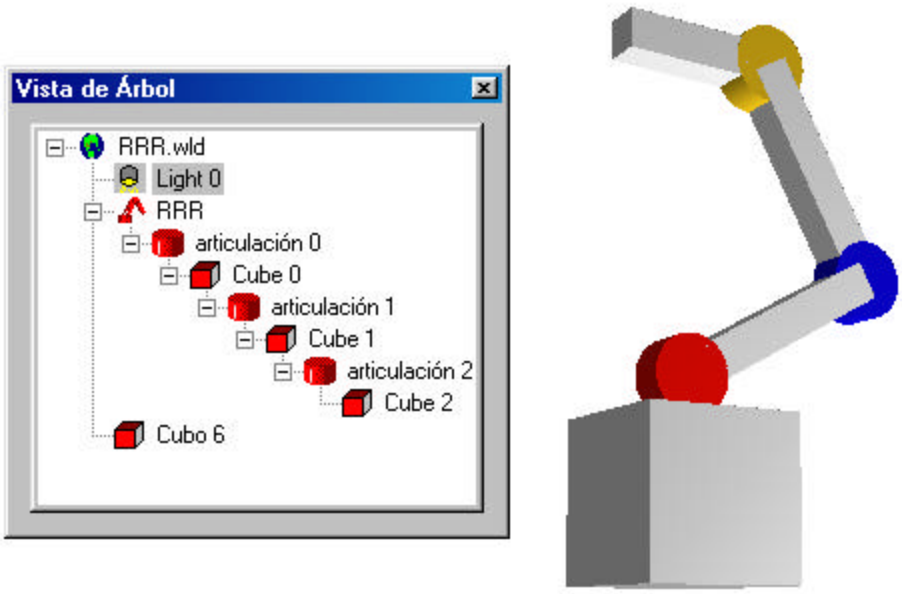


Figura 5. 9 Robot RRR

Era necesario también presentar distintos modos de ver la escena. Las opciones de visualización que se implementaron fueron las siguientes:

 ▪ Vista con perspectiva u ortogonal. Esta función únicamente modifica las propiedades de la cámara. Pueden llegar a ser muy útiles al momento de colocar los objetos en la posición deseada. La perspectiva es ideal para la vista isométrica, en cambio la ortogonal es ideal para la vista de frente o lateral.

- Seleccionar entre la vista de frente  , la lateral  , superior  o isométrica  .

 ▪ Mostrar los ejes de coordenadas. Esta opción permite ver en un momento dado los distintos ejes de coordenadas tanto del mundo como de los objetos.

- Agregar una cuadrícula  y la segunda opción es por si queremos que sirva de guía  .

- Vista de alambre  o sólido  .

5.2 Módulo de Control

Este módulo se encarga del control de los objetos y de los robots. Para este módulo se implementó un monitor de objetos para realizar cuatro operaciones fundamentales: mover, rotar, trasladar y escalar. Es posible además utilizar estas funciones por medio de una barra de comandos a través de la cual se selecciona la operación a realizar y se aplica directamente con el ratón en la pantalla. Las funciones de movimiento y rotación se pueden limitar a un rango por cada eje o inclusive bloquearse.

El monitor de objetos se muestra en la Figura 5.10, para modificar cualquier valor utilizamos las flechas a un costado de este. De esta manera incrementamos el valor o lo disminuimos. Los cambios los veremos reflejados automáticamente en la pantalla. Para bloquear o desbloquear un eje marcamos o desmarcamos la paloma junto a cada eje.

Para modificar un objeto es necesario primero seleccionarlo, la barra de selección nos permite determinar el objeto. La barra de selección presenta las siguientes opciones:

- Cámara.Las operaciones que realicemos se aplicarán a la cámara.

-  Objeto padre. Las operaciones que realicemos se aplicarán a los objetos padre.
-  Objeto hijo. Las operaciones que realicemos se aplicarán a los objetos hijos.
-  Eje. Las operaciones que realicemos se aplicarán al eje de los objetos.
-  Puntos. Las operaciones que realicemos se aplicarán los puntos en el mundo.

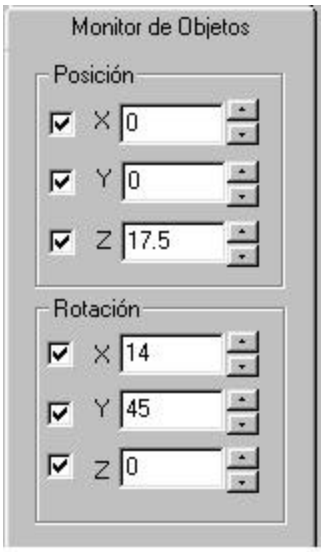


Figura 5. 10 Monitor de Objetos

Las barra de comandos nos permiten realizar las siguientes operaciones del elemento seleccionado:

-  Señalar objeto.
-  Mover objeto.
-  Rotar objeto.
-  Escalar objeto.
-  Bloquear /desbloquear eje X.
-  Bloquear/desbloquear eje Y.

-  Bloquear/desbloquear eje Z.

Se mencionó en la sección 5.1 de este capítulo que al momento de crear un objeto o de seleccionarlo aparecen los cuadros de diálogo de Origen, Rotación, Traslación y Escala. El diálogo de Origen (Figura 5.11) muestra la posición actual del objeto, junto con sus límites máximos y mínimos para cada eje. Si queremos cambiar estos valores solo tenemos que teclear los valores deseados.

Figura 5. 11 Diálogo de Origen

El diálogo de Rotación (Figura 5.12) muestra la rotación actual del objeto, junto con sus límites máximos y mínimos para cada eje. Si queremos cambiar estos valores solo tenemos que teclear los valores deseados. En la Figura 5.13 vemos el diálogo de Escala y Rotación que muestran los valores correspondientes para cada eje.

Existen dos problemas fundamentales a resolver en el caso del control del robot; el primero de ellos se conoce como el problema cinemático directo y consiste en determinar cuál es la posición y orientación del extremo final del robot con respecto a un sistema de coordenadas que se toma como referencia. Para este caso es necesario conocer el valor articular y las características geométricas de cada uno de los elementos del robot.

Figura 5. 12 Diálogo de Rotación.

El segundo problema se refiere a la cinemática inversa y consiste en resolver la configuración que debe adoptar el robot para una posición y orientación del extremo final del robot [Barrientos *et al*. 97]. Se tenían dos opciones, implementar un módulo o utilizar alguna librería ya existente. Se eligió la segunda opción debido al tiempo limitado del desarrollo del proyecto.

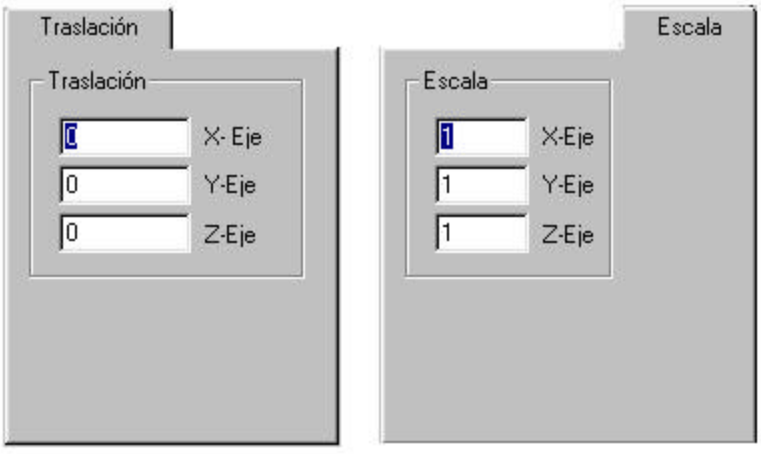


Figura 5. 13 Diálogos de Traslación y Escala

Se analizaron dos librerías, la primera de ellas fue "RGG Kinematix" del Robotics Research Group de la University de Texas en Austin a cargo de Ratheesh Rajan y Chetan Kapoor[]. La segunda fue ROBOOP de Richard Gourdeau[], trabajo presentado en la IEEE

of robotics. La librería más completa fue ROBOOP pero al mismo tiempo la más difícil de utilizar ya que requería de muchos parámetros para los robots no disponibles como el peso y centro de masa de sus articulaciones. En cambio la librería RGG Kinematix requería únicamente de la notación DH del robot en un archivo de texto para funcionar por lo que se opto por utilizar esta librería.

El monitor del robot muestra el valor de todas sus articulaciones y la opción de modificar su valor. El monitor del robot nos muestra también la posición del elemento terminal o Tool Center Point (TCP) como se le conoce comúnmente. Gracias a las funciones de cinemática inversa, es posible que desplacemos el elemento terminal del robot. Las articulaciones se moverán de manera que se alcance la posición deseada. En caso de que esto no sea posible, un mensaje de error aparecerá en la pantalla. Generalmente cuando ocurre este error es cuando el robot se encuentra en una singularidad. Las singularidades se clasifican en [Barrientos, *et al*. 97]:

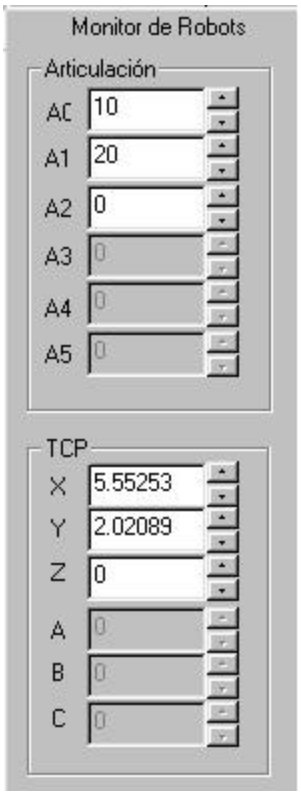


Figura 5. 14 Monitor de Robots

- Singularidad en los límites del espacio de trabajo del robot. Se presentan cuando el extremo del robot está en algún punto del límite de trabajo interior o exterior. En esta situación resulta obvio que el robot no podrá desplazarse en las direcciones que lo alejan de este espacio de trabajo.

- Singularidades en el interior del espacio de trabajo del robot. Ocurren dentro de la zona de trabajo y se producen generalmente por el alineamiento de dos o más ejes de las articulaciones del robot.

En caso de encontrarnos en este caso, tendremos que mover las articulaciones hasta "salir" de la singularidad. La barra de iconos del robot nos permite realizar dos operaciones muy útiles al momento de trabajar con un robot. La primera operación, es visualizar el TCP del robot. Esto lo hace desplegando una serie de puntos azules que nos indican por donde se

ha desplazado el elemento terminal del robot. La segunda operación todavía más útil nos permite mover el TCP en la pantalla con el movimiento del ratón. De esta manera podemos "indicarle" al robot el lugar donde queremos que coloque su elemento terminal.

5.3 Modulo de Programación

En el sistema ROBIN tenemos dos tipos de programación, la de los objeto y la de los robots. Programamos los objetos a través de almacenar su posición con respecto al tiempo. De esta manera el módulo de simulación repetirá estos pasos almacenados al tiempo correspondiente.

En el caso de los robots la programación es más estructurada. Un robot se puede programar igual que un objeto pero además puede tener un programa, el programa del robot esta compuesto de una serie de puntos. Cada punto contiene, el tipo de trayectoria con el que debe llegarse a él, el tiempo de llegada y el tiempo de espera en esa posición. Para este sistema se implementaron dos tipos de trayectorias, punto a punto (PTP) y lineal (LIN).

A continuación presento el ejemplo de un archivo PRG:

```
//  C:\Windows\Profiles\toño\Escritorio\Release\KUKA.prg
//  KUKA
//  < X, Y, Z, Tipo de Trayectoria, Tiempo de llegada, Tiempo de Espera >
Program Points {
         < 0.500000, -0.555019, 2.000000, LIN, 0.000000,  0.000000 >
         < 0.569377, -0.287421, 1.385519, PTP, 5.000000,  0.000000 >
         < 0.500000,  0.128844, 1.950440, LIN, 10.000000, 0.000000 >
}
```

La estructura de cada punto tiene la siguiente definición:

```
< X, Y, Z, Tipo de Trayectoria, Tiempo de llegada, Tiempo de Espera >
```

También existe una manera gráfica de crear el programa, esto lo podemos realizar al crear puntos en el mundo y asignándoselos al robot .

Cabe mencionar aquí la barra de íconos referente a los robots que vemos en la Figura 5.4. Esta barra permite:

- Asignar una serie de puntos como programa.
- Eliminar el programa.

- Visualizar el programa.
- Editar el programa como texto o como puntos.

Figura 5. 15 Barra de íconos de robot

5.4 Módulo de Simulación

Este módulo es el encargado de reproducir una secuencia de movimiento tanto de los objetos como de los robots. Para el caso de los objetos lo único que necesitamos hacer es, grabar su posición y orientación en un tiempo determinado y el sistema se encargará de reproducirlo. La animación de los objetos se realiza a través de una interpolación lineal de los atributos del objeto (posición, rotación) entre dos registros (ver Figura 5.17).

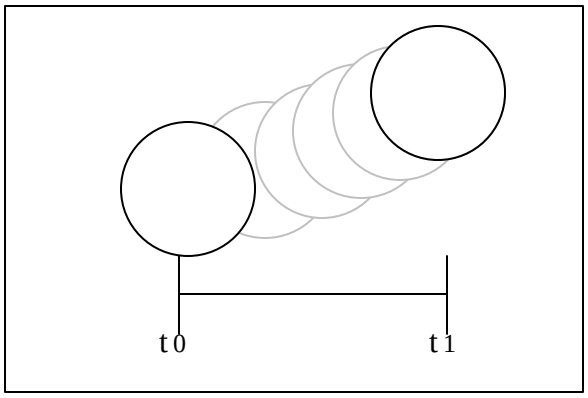


Figura 5. 16 Animación

Para el caso de los robots es posible también reproducir una secuencia de puntos contenida en un programa. Ya se presento en la sección anterior como es la estructura de este programa. La animación se realiza similar a la de los objetos en el caso de trayectorias punto a punto. En el caso de una trayectoria lineal, primero se calcula su posición entre dos puntos dados y después se utiliza la librería de cinemática inversa para asignar los valores de sus articulaciones correspondientes a ese punto.

Para realizar la simulación contamos con un control que contiene botones comúnmente utilizados. Podemos almacenar una posición, eliminarla, ir al principio o final de una serie de posiciones. Adelantar o retrasar una a una las posiciones. Un contador nos muestra en todo momento en que tiempo nos encontramos. Para ejecutar la simulación seleccionamos del menú Animar – Pantalla.

Figura 5. 17 Barra de iconos de animación

Capítulo 5 Implementación de ROBIN 50
5.1 Módulo Construcción 50
5.2 Módulo de Control 60
5.3 Modulo de Programación 65
5.4 Módulo de Simulación 66

Figura 5. 1 GLOOP ..... **¡Error! Marcador no definido.**
Figura 5. 2 Parámetros D-H para un eslabón giratorio 57
Figura 5. 3 Notación DH para un robot RRR. 57
Figura 5. 4 Barra de íconos de robot 66
Figura 5. 5 Barra de iconos de animación 67