SIMULACIÓN Y CONTROL DE UN ROBOT UNIMATE PUMA 200

F. Porras S., °A. Muñoz I., F. Reyes C., J. Cid M.,
J. Méndez M., G. Villegas R., * A Lara E.
Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Electrónica, Grupo de Robótica y Control. *Departamento de Microelectrónica Instituto de Ciencias. °Facultad de Ciencias de la Computación.
18 sur y San Claudio, Col. San Manuel, C. U. Edif. 129
CP. 72570, Puebla, Méx. Tel: (2) 2 295500 Ext. 7410, 7405 Fax: (7400).
Email: fporras@ece.buap.mx : alexmun@ixtchel.cs.buap.mx : jmm@ece.buap.mx

RESUMEN

El trabajo que se presenta a continuación es la implementación previa a un sistema de simulación y control para un Robot UNIMATE PUMA 200. Para el desarrollo del presente programa, fue utilizada la metodología de la programación orientada a objetos, realizando la codificación del programa por medio del lenguaje C++ y para su implementación se utilizó el lenguaje visual Borland C++Builder ver 5.0.

Con la ayuda de AutoCAD 2002 se creó el diseño de la estructura del Robot PUMA, que consistió en construir para cada pieza del Robot su correspondiente representación a nivel de malla (solamente líneas). A partir de estos modelos, se elaboraró su respectiva representación en 3D, usando para el desarrollo de estas simulaciones, las librerías del lenguaje C que provee OpenGL™.

OpenGL™ han sido elaborado por Silicon Graphics Inc. desde 1992, la cual proporciona librerías de programación para varios lenguajes de alto nivel, como también su uso en múltiples sistemas operativos. Con la ayuda de OpenGL™, se crea una interfaz directa entre la aplicación y la tarjeta de video de la computadora, dándole a esta mayor velocidad de acceso al momento de dibujar un entorno gráfico en la aplicación.

1. INTRODUCCION

La empresa UNIMATE produce una amplia gama de Robots industriales entre los cuales, se encuentra la línea de Robots PUMA 200. Las características de estos robots varían según usos y aplicaciones. Los sistemas de control de estos robots son de arquitectura cerrada, por lo que la industria se ve forzada a comprar diferentes modelos dependiendo de la aplicación y con ello elevando sus costos en su adquisición y mantenimiento.

El proyecto desarrollado inicialmente por el Laboratorio de Automatización, Robótica y Control de la Facultad de Ciencias de la Electrónica de la Benemérita Universidad Autónoma de Puebla fue la sustitución del sistema de control original de un Robot PUMA 200, por un nuevo sistema de control construido a base de tarjetas de adquisición de datos, conservando solamente del sistema original la etapa de potencia del Robot.

La sustitución exitosa del sistema de control por el nuevo sistema trajo consigo la reducción de costos como: el mantenimiento del sistema y el reemplazo de piezas dañadas. Esto condujo a que cualquier persona calificada con conocimientos sobre el funcionamiento del sistema de control, pueda efectuar la reparación y mantenimiento de este sin ningún problema.

El reemplazo total del sistema de control original del Robot PUMA trajo consigo, que el lenguaje de programación con el que trabajaba el Robot quedara inservible, ya que este utilizaba como interfaz el viejo sistema de control. Por lo tanto, un objetivo primordial después de la sustitución del sistema de control, es la creación y adaptación de un sistema de programación y simulación con un entorno grafico 3D para el control y manejo del Robot PUMA.

2. DESARROLLO

2.1 Técnicas de Modelado de sólidos en 3D

La técnica de representación por frontera es la más utilizada para realizar la representación de un objeto grafico tridimensional. Esta técnica consiste en definir la estructura del objeto a través de una superficie formada

por un conjunto de polígonos, los cuales comparten sus aristas y vértices. [2]

Una organización conveniente para almacenar los datos geométricos de un conjunto de polígonos, es crear tres listas: una lista de vértices, una de aristas (lados) y otra de polígonos (caras). [2]

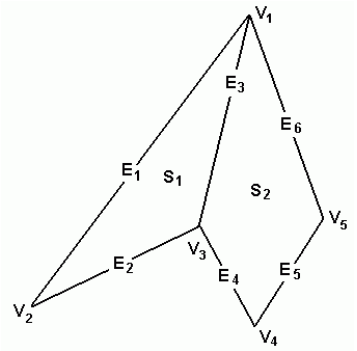

Figura 1: Representación de una superficie por medio de dos polígonos adyacentes

A continuación se presentan las tablas correspondientes a la figura anterior, las cuales representan a dos polígonos adyacentes, formados a partir de seis aristas y cinco vértices.

Vértices
V_1: x_1, y_1,z_1
V_2: x_2, y_2,z_2
V_3: x_3, y_3,z_3
V_4: x_4, y_4,z_4
V_5: x_5, y_5,z_5

Tabla 1: Tabla de Vértices de la Figura 1

Aritas
E_1: V_1,V_2
E_2: V_2,V_3
E_3: V_3,V_1
E_4: V_3,V_4
E_5: V_4,V_5
E_6: V_5,V_1

Tabla 2: Tabla de Aritas de la Figura 1

Caras
S_1: E_1, E_2,E_3
S_2: E_3, E_4,E_5, E_6

Tabla 3: Tabla de Caras de la Figura 1

La ventaja de las representaciones por medio de polígonos, es que son más adecuadas para la visualización utilizando el hardware gráfico que trabaja con polígonos. El problema que plantea este tipo de objetos, es que la mayoría de estos no son realmente poliedros, sino que están compuestos por superficies curvas continuas. [1]

Por ejemplo: para la representación de un objeto complejo se puede utilizar una descomposición de su superficie secciones triangulares o cuadrangulares. [1]

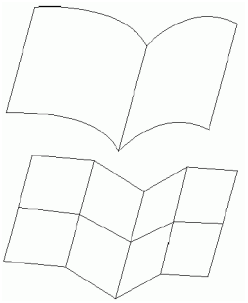

Figura 2: Representación de una superficie curva por medio de una malla de polígonos

2.2 Objetos y Clases básicas

El uso de la programación orientada a objetos puede reducir sustancialmente el esfuerzo y tiempo que se ocupa para el desarrollo y mantenimiento de grandes sistemas para computadoras. [3]

En contraste con la programación estructurada, los lenguajes orientados a objetos se concentran en el diseño de objetos o clases, los cuales encapsulan una idea en particular o una construcción matemática del problema. El encapsulado de los datos se refiere a que estos pueden ser manipulados sólo a través de una interfaz bien definida de la clase. El programador o usuario que utiliza una clase, no tiene que preocuparse por la organización de los datos y de cómo se llevan a cabo las operaciones individuales, por lo cual es suficiente dominar la interfaz de la clase, la cual frecuentemente puede ser descrita de forma breve, de manera que la documentación de la clase se simplifica en gran medida. [3]

A continuación se realizará una descripción general de los objetos que ponen al sistema de simulación grafica en 3D del Robot PUMA.

Podemos afirmar que un objeto geométrico del tipo que se requiere para modelar el PUMA puede estar conformado sólo por dos tipos de objetos: vértices y caras. Un vértice es un tipo de dato que guarda básicamente tres valores (x, y, z), los cuales representan un punto en el espacio tridimensional *OXYZ*, donde el componente *x* representa lo ancho, el componente *y* lo largo y el componente *z* representa la altura. Una cara o parche es simplemente la representación de un polígono, el cual esta formado con al

menos tres vértices o puntos en un espacio dimensional o tridimensional, este polígono básico se conoce como: triángulo.

Un entorno grafico en 3D se puede comparar con un escenario donde hay actores, escenografía, luces, etc. Y al menos se cuenta con una cámara teniendo esta la función principal de visualizar el interior del escenario.

Un escenario puede estar al aire libre, parcial o completamente cerrado, puede no tener ninguna luz o contar con una hasta varias luces. Tiene un piso o área que indica sus límites, los cuales comúnmente están marginados por paredes o por una rejilla.

Los actores y la escenografía son objetos tridimensionales que se encuentran dentro del escenario, los cuales son visualizados por medio de una cámara. Ladiferencia entre ellos es que los actores pueden realizar movimientos y acciones que afectan a la escenografía y a otros actores. En cambio la escenografía no puede moverse a menos que un actor realice una acción como por ejemplo: mover un objeto de su lugar, etc.

Una cámara dentro de un escenario tiene la función de realizar la visualización del interior de escenario, como también a los actores y a la escenografía. Inicia en una cierta posición, puede enfocar a objeto en particular o a varios al mismo tiempo, puede cambiar el punto de visualización. Realiza movimientos como avanzar, retroceder, ir a la izquierda o a la derecha con respecto de su propio lugar o a partir de la posición de algún objeto. También puede realizar rotaciones (giros) alrededor de una posición específica o sobre de sí misma, etc.

Una luz cuenta con una posición pudiendo cambiar ésta si fuera necesario. Puede tener un determinado color con cierta intensidad, aparte de diversos factores que afectan de manera directa la percepción de los materiales con los que están constituidos los objetos del escenario.

Tomando las anteriores descripciones de los objetos encontrados en el ambiente de un entrono de simulación 3D, se realizó el correspondiente análisis orientado a objetos y para la fase de instrumentación del diseño se utilizó el lenguaje C++ con las librerías del lenguaje C que ofrece OpenGL.

A continuación se mencionan y describen de manera general las Clases básicas que se obtuvieron al finalizar el análisis y diseño orientado a objetos:

TVector3D: *Clase que genera un objeto vector en R^3, los objetos de esta clase tiene propiedades que pueden guardar coordenadas (x, y, z).*

TCara: *Clase que genera y guarda los datos básicos para representar un triángulo. Contiene principalmente los índices de los tres vértices que forman al triángulo, el vector normal del plano y el índice del material con el que se dibuja el triángulo.*

TEscenario: *Clase que genera un objeto que gestiona e inicializa el entorno gráfico de OpenGL en la aplicación.*

TMaterial: *Clase que contiene las propiedades necesarias para guardar la declaración de un tipo de material.*

TObjeto3D: *Clase que contiene y genera los objetos necesarios para guarda la representación de un objeto tridimensional. Contiene a las clases: TVector3D, TCara y TMaterial.*

Las clases TCamara y TLuzGL: *Están en proceso de programación.*

2.3 El AutoCAD 2002 como herramienta fundamental de diseño.

Para la creación de una representación de un objeto tridimensional se deben de obtener todos los vértices que conforman al objeto, ya que a partir de estos se construye el conjunto de polígonos que unidos entre si forman a dicho objeto.

El problema que se presentó al construir cada una de las piezas que componen al Robot PUMA, fue que no eran simples de construir con figuras geométricas sencillas, como: triángulos, cuadrados, cubos, trapecios o alguna otra figura geométrica básica. Ya que a simple vista la estructura del Robot está compuesta por figuras geométricas como: cilindros y/o esferas, obtener la lista total de vértices de cada pieza del Robot no iba a ser tarea sencilla.

Además, de que una de las especificaciones que se pretendía alcanzar para el sistema de simulación, era la representación detalla de la estructura real del PUMA, por lo cual quedo descartada inicialmente la opción de realizar una representación burda del Robot.

Por lo tanto para la construcción de los diseños a nivel de líneas, fue elaborada con la ayuda de AutoCAD 2002, donde se pudo obtener de manera simple las coordenadas (x, y, z) de cada vértice que forma parte de la estructura de dicha pieza del Robot.

A continuación se presentan los diseños hechos con AutoCAD 2002 de dos piezas del Robot PUMA.

Imagen 1: Representación de la pieza del Robot PUMA Autodenominada: Hombro

Imagen 2: Representación de una pieza del Robot PUMA Autodenominada: Antebrazo

Al mismo tiempo que se iban obteniendo las coordenadas de cada uno de los vértices del objeto tridimensional, estos datos se iban guardando en un archivo de texto con extensión: OGD (Object Geometric Data) donde también se iban formando y guardando la lista de triángulos que generan al objeto 3D.

A continuación, se muestra una tabla con los datos de las primeras cinco piezas que forman la simulación 3D del Robot PUMA 200:

Pieza	# Vértices	# Caras	# Materiales
Base	353	654	1
Hombro	301	590	1
Antebrazo	311	519	4
Brazo	302	582	3
Base Muñeca	160	347	1

Tabla 4: Tabla comparativa entre los datos de cada archivo OGD

2.4 Pruebas y Resultados

Los aspectos que se tomaron en cuenta para probar el funcionamiento correcto del programa de simulación 3D, son los siguientes:

- Traslaciones y rotaciones de la cámara
- Rotaciones de toda la estructura del Robot
- Movimientos independientes de cada eslabón que contiene hasta ahora la simulación.

A continuación, se mencionan algunas de las computadoras en las que fue probado el programa de simulación 3D:

Computadora 1:
Microprocesador: AMD Athlon a 1.8 Ghz
Memoria RAM: 128 MB
Disco Duro: 60 GB
Tarjeta de video: *ATI Rage 128 GL Pro a 32MB*
Sistema Operativo: Windows 2000 Profesional

Computadora 2:
Microprocesador: AMD K6-2 3D Now a 300 Mhz
Memoria RAM: 160 MB
Disco Duro: 6.4 GB
Tarjeta de video: *SIS 6326 a 8MB*
Sistema Operativo: Windows ME

Computadora 3:
Microprocesador: Pentium IV a 2.0 Ghz
Memoria RAM: 256 MB
Disco Duro: 60 GB
Tarjeta de video: *eTNT Evidia Gforce a 32MB*
Sistema Operativo: Windows XP Profesional

Computadora 4:
Microprocesador: Pentium II a 500 Mhz
Memoria RAM: 256 MB
Disco Duro: 8 GB
Tarjeta de video: *S3 a 8MB*
Sistema Operativo: Windows NT WorkStation 4.0

Las siguientes imágenes muestran la ejecución del programa de simulación 3D del Robot PUMA 200.

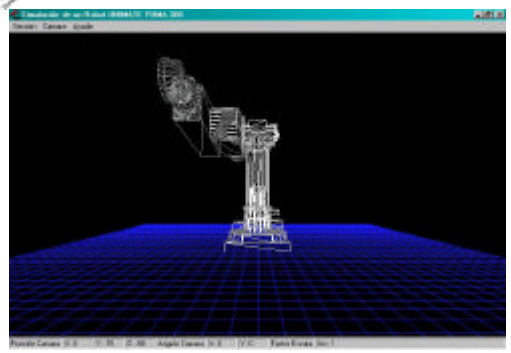

Imagen 3: Posición inicial del Robot
(Modelo a Nivel de Líneas)

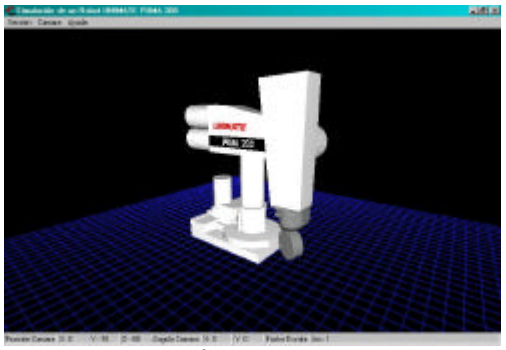

Imagen 4: Visualización del Robot PUMA con diversas rotaciones de sus eslabones. (Modelo a Nivel de Polígonos)

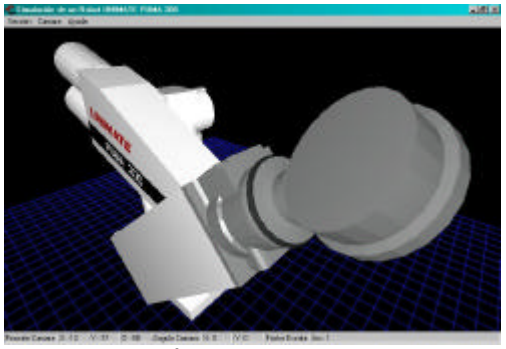

Imagen 5: Visualización del Robot PUMA con traslaciones y rotaciones de cámara, además de algunas rotaciones de sus eslabones. (Modelo a Nivel de Polígonos)

3. CONCLUSIONES

Como prototipo previo al sistema de simulación 3D para el Robot Industrial PUMA 200 se probó que, el desarrollo e implementación del entorno gráfico tridimensional por medio de OpenGL es factible para la elaboración de sistemas con entornos tridimensionales, sin tener que comprar tarjetas de video potentes a costos elevados pudiendo utilizar tarjetas de video de mediana calidad y desempeño.

Se comprobó que la metodología de programación orientada a objetos facilitó el desarrollo e implementación de la aplicación, ya que en ves de declarar múltiples variables que guarden toda la geometría y topología de un objeto tridimensional, sólo fue necesario crear objetos de la clase TObjeto3D.

Se verificó y comprobó que el desempeño de adaptar los tipos y funciones que ofrece OpenGL por medio de sus librerías de C con el lenguaje C++ es fiable, como también el uso de OpenGL en Borland C++Builder 5.0.

Las futuras mejoras al presente proyecto, serán:

- Complementar las clases que componen e interactúan en el entorno grafico.
- Realización de una implementación básica de las librerías y funciones que provee OpenGL. Ya que esta API (Application Programming Interface) ofrece mucho más potencia de desarrollo para graficado por computadora.

Este programa como ya se mencionó, es el desarrollo previo a todo un sistema de simulación y control de un Robot UNIMATE PUMA 200 en un entorno 3D donde se pueda realizar simulaciones visuales previas del trabajo asignado y sus correspondientes depuraciones con el robot off-line.

REFERENCIAS

[1] Silicon Graphics Inc, *OpenGL Programming Guide* (United States of America, Addison-Wesley Publishing Company, 1997)

[2] Donald Hearn, M. Pauline Baker, *Gráficas por Computadora*. Segunda Edición. (Editorial Prentice Hall)

[3] Roger S. Pressman, *Ingeniería del Software. Un enfoque práctico*. Cuarta Edición. (Madrid, España. Editorial McGraw-Hill,1997)