NAVEGACIÓN DE UN ROBOT MOVIL UTILIZANDO GRAFICACIÓN VIRTUAL TRIDIMENSIONAL

R. Osorio, J. Silva, A. Arteaga, M. Peña, * J. Savage.

Departamento de Ingeniería de Sistemas Computacionales y Automatización.
Sección de Electrónica y Automatización.
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
Apdo. Postal 20-726 , México D.F.
Tel/fax: 5616-0176
Email: roman@servidor.unam.mx
* Facultad de Ingeniería., UNAM.
Laboratorio de Interfases Inteligentes

Resumen

La constatación de la dificultad de programar un robot para operaciones complejas y con capacidad de adaptación a situaciones cambiantes ha hecho resurgir la idea de la tele-operación. Recordemos que la tele-operación consiste en el manejo de un robot a distancia el cual obedece a las instrucciones que proporciona el usuario ya sea a través de otro robot o un sistema de cómputo que las genere; la tele-operación permite el manejo del robot remoto con una mayor facilidad si este proporciona algún dato sobre su estado ya sea a través de una imagen o coordenadas que lo ubiquen en un entorno. Otro factor es el costo, para estas aplicaciones es muy alto ya que en la actualidad existen robots tele-operados, pero esta tele-operación se da por medio de una plataforma especializada, lo que requiere que en la máquina en donde se desee hacer la tele-operación tenga este sistema operativo.

Es por esto, que el objetivo del trabajo que se presenta tiene como finalidad generar un software en un ambiente de realidad virtual, que permita realizar la navegación del robot móvil (se cuenta con un prototipo) durante su operación en un ambiente de alto riesgo para el personal de una planta de energía núcleo eléctrica, o bien cualquier ambiente industrial que represente un riego para el humano, esto se desarrollara por medio de la creación de ambientes tridimensionales cambiantes, bajo una plataforma muy usual como lo es Windows. Además de que el robot virtual pueda interactuar con un ambiente prediseñado y al mismo tiempo pueda ser tele-operado a un costo relativamente bajo.

Palabras Clave: Robot Móvil, Realidad Virtual, Graficación tridimensional.

1. INTRODUCCION

HERRAMIENTAS DE DESARROLLO PARA VIDEOJUEGOS

Para el desarrollo de este articulo es importante conocer las herramientas existentes que se están utilizando en la creación de videojuegos, ya que en este entendimiento se puede dar una introducción a los ambientes tridimensionales que dará el panorama general para lograr la elección adecuada de las herramientas de desarrollo. De esta forma a continuación se presentan en primer lugar las herramientas más conocidas para el desarrollo de videojuegos:

DIV Game Studio

DIV es un lenguaje de programación especialmente diseñado para videojuegos, aunque se pueden programar todo tipo de aplicaciones. DIV fue creado por Daniel Navarro como proyecto de fin de carrera.

Fénix

Es un lenguaje pseudo-interpretado diseñado para programar juegos 2D. Para ello incluye una extensa librería dedicada a los juegos que permite programar únicamente la lógica de movimientos de los gráficos y objetos del juegos, mientras cosas como la visualización en pantalla de gráficos y sprites, o la reproducción del sonido, corren a cargo del interprete incluido.

BlitzBasic

BlitzBasic es un compilador de juegos 2D que utiliza DirectX 7. Está basado en el lenguaje Basic(aquí sólo se manejan gráficos 2D). Sus características principales son:

DarkBasic

Es un lenguaje de alto nivel que como su nombre lo indica está basado en el lenguaje Basic, sólo que esta versión está basada en DirectX y orientada a la creación de videojuegos en 3D.

La plataforma para que compile DarkBasic es Windows, aunque en realidad no compile binarios, sino, como Div o Fénix, crea un código intermedio que después es interpretado en tiempo de ejecución, siendo esto algo más lento que un programa compilado a código máquina.

Recordemos que el objetivo es tener ambientes gráficos modificables para correr en plataforma Windows y desarrollar el software en Visual Basic (VB). Por lo que las anteriores herramientas para el desarrollo de videojuegos no son muy apropiadas, ya que las dos últimas corren en Windows pero no cumplen con la limitante de programarse bajo VB.

SELECCIÓN DEL LENGUAJE DE PROGRAMACION

Utilización de Visual Basic(VB)

Hablando del rendimiento que se puede obtener utilizando VB en comparación con C y C++ se puede asegurar que los programas se ejecutan casi a la misma velocidad, alrededor del 10% de pérdida en rendimiento, siempre dependiendo de la configuración final que se le asigne al ejecutable a la hora de compilarlo.

No existe una respuesta definida para determinar cuál es el mejor lenguaje para programar ambientes gráficos ni qué API gráfica es la mejor. Cada lenguaje y API tiene sus propias características con un enfoque propio para resolver problemas; decidir si es bueno o malo es relativo a lo que se busca o el tipo de problema con el que se cuente.

Después de haber analizado los diferentes lenguajes de programación que existen debemos recordar que por especificaciones del problema el lenguaje de programación que se utilizara es Visual Basic, ya que por motivos de compatibilidad con otros programas ya desarrollados es necesaria la utilización de este lenguaje. Con esto ya se tiene resuelto el problema del lenguaje y nos da pie para el análisis del API a utilizar.

SOFTWARE DE GRAFICACIÓN

En el desarrollo de ambientes gráficos tridimensionales normalmente se utilizan aplicaciones de apoyo que permiten el diseño externo de objetos complejos que formarán parte de la aplicación ya sea ésta un video juego o alguna aplicación multimedia, para lo cuál, en última instancia, deberán importarse desde la aplicación que las utilizará. Así, se debe elegir algún software de graficación que nos asista en la creación de los objetos tridimensionales.

Existen numerosos y conocidos programas en el mercado para crear imágenes (Fractal Design Painter, LigthWave 3D, 3D Studio MAX, Photoshop, Corel Draw, AutoCAD,...), por mencionar algunos. Habiendo mucho software gráfico con el cual podemos interactuar haremos una selección de acuerdo a los requerimientos de nuestro problema que sería el trabajo con Windows y compatibilidad con Visual Basic comunicándose a través de DirectX. Es por ello que sólo mencionaremos a aquellos que parecen ser los más utilizados en el área.

HERRAMIENTA ACTIVEX

ActiveX es el nombre que Microsoft ha dado a un grupo de tecnologías y herramientas "estratégicas" orientadas a objetos. Su principal tecnología es el Modelo de Objeto Componente (*Component Object Model, COM*). Al usarlo en una red con un directorio y apoyo adicional, el COM se convierte en el Modelo Distribuido de Objetos Componentes (*Distributed Component Object Model, DCOM*). El principal objeto que uno crea al escribir un programa ejecutable en el entorno ActiveX es un componente, un programa autosuficiente que puede ejecutarse en cualquier sitio en la red ActiveX (que es actualmente una red que consta de sistemas tanto Windows como Macintosh). Este componente se conoce como un Control ActiveX. ActiveX es la respuesta de Microsoft a la tecnología Java de Sun Microsystems. Un control ActiveX es aproximadamente el equivalente a un applet Java.

El componente ActiveX

Las grandes categorías de eventos que Microsoft recomienda tener en consideración para definir los eventos característicos de los componentes ActiveX son las siguientes:

1. Las peticiones
2. Los eventos preoperativos (before events)
3. Los eventos postoperativos (after events)
4. Los eventos operativos (do events)

Las peticiones son eventos enviados por el componente ActiveX al documento para autorizarlo a proseguir o emprender una operación. Por ejemplo, un componente ActiveX que hubiera recibido un clic sobre el botón Cerrar, puede enviar una petición al documento para autorizar su cierre.

2. REALIDAD VIRTUAL

La realidad virtual es un área que combina distintas tecnologías para visualizar, manipular e interactuar con una computadora en una forma que busca simular comportamientos y percepciones naturales para el hombre. La Realidad Virtual es la tecnología que permite sumergir a un usuario en un ambiente tridimensional simulado por el computador, de forma interactiva y autónoma en tiempo real.

Inmersión

Esta palabra significa bloquear toda distracción y enfocarse selectivamente solo en la información u operación sobre la cual se trabaja. Posee dos atributos importantes, el primero de ellos es su habilidad para enfocar la atención del usuario, y el segundo es que convierte una base de datos en experiencias, estimulando de esta manera el sistema natural de aprendizaje humano (las experiencias personales).

Tridimensionalidad

Esta es una característica básica para cualquier sistema llamado de realidad virtual, tiene que ver directamente con la manipulación de los sentidos del usuario, principalmente la visión, para dar forma al espacio virtual; los componentes del mundo virtual se muestran al usuario en las tres dimensiones del mundo real, en el sentido del espacio que ocupan, y los sonidos tienen efectos estereofónicos (direccionalidad).

3. DESARROLLO

ALGORITMO PARA LA DETECCIÓN DE COLISIONES

La detección de colisiones ha sido un problema fundamental en la animación por computadora, modelado físico, modelado geométrico y robótica. En esas aplicaciones la interacción entre objetos que se mueven es modelada con restricciones dinámicas y análisis de contacto. El movimiento de los objetos está restringido por varias causas incluyendo colisiones.

Implantación del algoritmo de detección de colisiones

Debido a que nuestro ambiente se encuentra postrado en un solo plano (el piso) y el único objeto que se está moviendo es el robot movil, entonces podemos implantar un algoritmo de detección de colisiones por círculos el cuál pasará al siguiente nivel de prueba de colisión si los círculos se traslapan.

El siguiente nivel de detección de colisiones aprovecha las utilidades desarrolladas por la librería D3DX8 de DirectX la cual proporciona la prueba de cajas envolventes. Esta función determina si una raya (ray) interseca a la caja envolvente de un modelo. Así se especifican las rayas de un modelo y se comparan con la caja envolvente del otro modelo a través de la función que proporciona DirectX.

Debido a que este algoritmo no permite una detección precisa de la colisión con objetos de formas complejas, nos vimos en la necesidad de implantar un método alternativo para el usuario el cual sustituye la detección por cajas y prueba la malla contra una raya. Así, nos apoyamos en otra función de DirectX que necesitas como parámetros esenciales una malla y una raya y determina si la raya intersecta a la malla. El método preciso toma cada una de las rayas que conforman la malla de un objeto y las compara con la malla del otro objeto.

Después de haber implantado los dos algoritmos pudimos observar que el primer algoritmo es realmente eficiente cuando en el ambiente no se encuentran objetos irregulares o con espacios debajo de ellos como puede ser una mesa sencilla debido a que el espacio que se encuentra debajo de la mesa es parte de la caja envolvente del objeto. El segundo algoritmo disminuye en gran medida la velocidad con que se puede mover la sonda robot ya que toma mucho más tiempo de procesamiento el método de detección de colisiones exacto.

Cabe mencionar que dentro de los ambientes que se pretenden manejar con esta aplicación únicamente se considera al carrito (sonda robot) como el objeto en movimiento, es decir, se considera al ambiente estático. Por otra parte, los objetos que se encuentran en el ambiente están al piso y se espera sean de formas simples (cajas, esferas, cubos, etcétera) por lo que se considera el primer algoritmo como suficiente para satisfacer las necesidades de la aplicación. Se deja como opción el algoritmo preciso para que el usuario pueda interactuar con los espacios no accesibles que considera el primer algoritmo.

DESCRIPCIÓN DEL DESARROLLO DEL SOFTWARE

Esta parte está desarrollada con la finalidad de describir las funciones más importantes de la aplicación que se diseñó con el apoyo del marco teórico ya expuesto; principalmente se mostrarán las funciones codificadas que tienen su fundamento en la explicación teórica o que fueron determinantes para el desarrollo de la aplicación.

a) Obtención, Configuracióny Dibujado(Renderización) del dispositivo DirectGraphics

Para desarrollar la aplicación existen dos formas en pantalla completa o modo ventana; se eligió esta última debido a que era necesario interactuar con la aplicación a través de botones y cajas de texto además de facilitar la configuración del dispositivo adaptándose al modo actual de video del sistema que lo esté ejecutando. De esta forma al iniciar la aplicación una de las primeras cosas que se crean es el dispositivo gráfico sobre el que se va a trabajar. Las funciones que se encargan de inicializar el dispositivo de video son ***Inicializacion, Obtener_Disp.*** Inicialización obtiene información del estado actual del monitor y verifica la capacidad del acelerador gráfico para la graficación de la profundidad(Z- Buffer), además de inicializar las variables necesarias para la ejecución del programa.

La función ***Obtener_Disp*** se encarga de crear y asociar un dispositivo al picture1 además de determinar el tipo de texturizado que se utiliza, así como el tipo de luz y el modo de procesamiento de vértices.

b) Inicialización del Ambiente

Una parte muy importante en el desarrollo de la aplicación además de la creación del dispositivo gráfico es la definición del sistema de coordenadas, el punto en donde se ubicará la cámara y la perspectiva que se tendrá del ambiente; estos tres aspectos se cubren al asignar la matriz correspondiente al dispositivo, con esto se define como se verá la imagen en la pantalla. Esto se puede analizar en la función de ***Iniciar_Vista*** la cual se muestra a continuación:

Además, aunque no esenciales, existen un par de funciones que se encargan de orientar al usuario dentro del ambiente las cuales son ***inicia_plano e Iniciar_cuadricula. inicia_plano*** carga una textura al dispositivo y la muestra como el piso de referencia. Por otra parte ***Iniciar_cuadricula*** tiene la misma función de proporcionar la referencia del piso pero a través de una cuadricula.

c) Carga de los archivos .X

Para importar las geometrías prediseñadas en alguna otra aplicación de diseño tridimensional y aprovecharlas en el desarrollo de nuestro ambiente, recurrimos a funciones de DirectX específicamente a las que proporciona la librería **D3DX8** las cuales permiten la carga de archivos en formato .X, obteniendo información sobre la geometría prediseñada tal como es el tipo de material y textura de cada uno de los subgrupos que integran la geometría importada. La función que se encarga de realizar esta tarea es ***Crear_Nuevo_Objeto***; además de cargar archivos, esta función llena una estructura de datos que permite inicializar adecuadamente el objeto en la aplicación, esta estructura contiene información referente a su posición, rotación, dirección, centro, coordenadas máximas y mínimas de su caja envolvente, datos que permiten que la aplicación manipule al objeto

d) Métodos de inserción

Existen dos formas de agregar los objetos al ambiente, una más precisa(teclado) que la otra(mouse). La inserción por teclado tiene la desventaja de que a pesar de ser un método exacto el usuario tiene que estar muy familiarizado con el ambiente y el tipo de objetos que desee insertar pero sobre todo con el sistema de coordenadas que se utiliza en esta aplicación.

El método de inserción por teclado se programó con dos cajas de texto modificables por el usuario las cuales establecen las coordenadas(X, Z) donde se encontrará el centro del objeto a insertar.

e) Guardado y Apertura del ambiente

Para conservar los ambientes generados previamente se desarrollaron dos funciones encargadas de guardar y abrir los archivos, así como la definición de la organización e interpretación de los datos dentro de los archivos. Debido a la complejidad geométrica posible de un ambiente se optó por guardar en un archivo de texto las posiciones de los objetos finales dentro del ambiente además de una referencia al archivo que contiene la figura prediseñada. Por cada objeto encontrado en el ambiente se crea una copia del archivo que lo contiene y se traslada a una carpeta generada previamente en la misma ruta en donde se desea guardar el ambiente. Junto con estos datos se guarda la dimensión del plano, pero no se guarda la posición final del robot, esto con la finalidad de que éste pueda ser insertado en otra parte del ambiente diseñado cuando se vuelva a cargar el ambiente.

Para la función de ***Apertura*** se lee el archivo de texto y se cargan los objetos geométricos referenciados y se buscan en la carpeta con el mismo nombre del archivo que se debe encontrar en la misma ruta. Teniendo los datos de ese archivo de texto, los objetos cargados se transladan a las coordenadas correspondientes y se cargan tanto el robot real como el robot virtual esperando que se dé el click para su posición de inserción

f) Métodos de Conducción

Traslación

Como se mencionó anteriormente, para animar un objeto existen dos operaciones básicas: translación y rotación; en este caso, la translación del objeto se logra extrayendo

sus vértices, se aplica el incremento o decremento en sus coordenadas y se actualiza el objeto geométrico.

g) Detección de colisiones

Después de analizar los métodos para detección de colisiones se encontró que, si bien en este caso se podría necesitar de un método exacto para simular realidad en la colisión, el sistema necesita mucho tiempo de procesamiento debido a sus característica; sin embargo al contar con sensores en el robot se pudo percibir que una colisión se detecta a una distancia considerable respecto del objeto, por lo que, para simular ese comportamiento, se puede utilizar el método de cajas envolventes que necesita un menor grado de procesamiento. Si no se quiere simular ese comportamiento se encontró que la librería D3DX8 tiene desarrollada una función que detecta la intersección de una raya y una malla(que es un conjunto de triángulos). Al final se optó por implantar un sistema de detección de colisiones incremental el cual en primera instancia detecta la intersección de los círculos que rodean a cada objeto y si existe una intersección, de acuerdo con el algoritmo elegido, se prueba la caja circundante o la intersección exacta entre los objetos.

h) Interacción con el Robot Real

Comunicación serie

Para implementar la comunicación serie se optó por utilizar un control ActiveX que viene incluido como componente Visual Basic. Para poder establecer una comunicación a través de un puerto únicamente se necesita asegurar que el puerto en cuestión se encuentre cerrado, establecer los parámentros de comunicación y abrir el puerto para poder empezar a enviar y recibir datos. El tipo de comunicación que se puede establecer a través del puerto es por un ciclo cerrado o por eventos. En este caso se optó por la comunicación por eventos tratando de evitar la pérdida de datos.

Ajustes entre imagen real y virtual

Se está consciente que la simulación del movimiento del robot real y el movimiento del robot virtual diferirán en algún momento e inclusive en magnitudes posiblemente no aceptables debido a las imprecisiones mecánicas del robot, por lo que la precisión del ajuste en las pruebas realizadas determinan las relaciones avance/instrucción y grados/instrucción y con ello la calidad de la simulación. Es por lo anterior que necesitamos proporcionar al usuario una forma de llevar el robot virtual al lugar dónde se encuentra el robot real en el momento que las posiciones de ambos difieran en gran medida

4. CONCLUSIONES

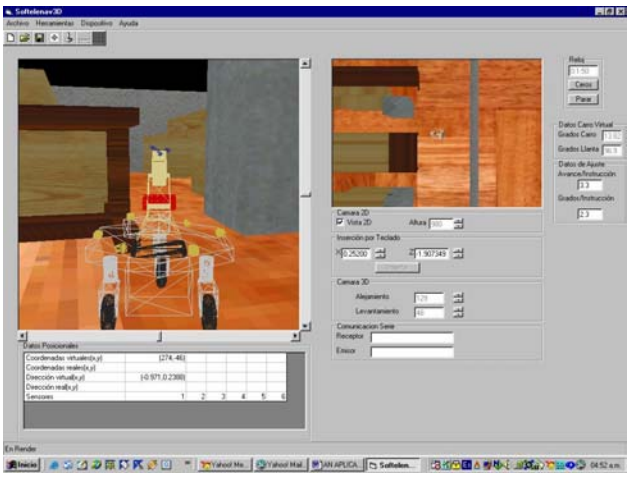
Se concluye que se ha terminado una aplicación que permite manejar a distancia un robot que se mueve en ambientes estáticos, que pueden ser peligrosos o inaccesibles para el ser humano como pueden ser radiación, electricidad de alta potencia, temperaturas extremas, gases peligrosos, espacios pequeños, etcétera. A través de esta tele-operación se puede arriesgar sólo al robot dentro de esos ambientes hostiles logrando que el operador quede fuera de cualquier peligro pero aportando su destreza en la conducción y en toma de decisiones.

Las ventajas principales que se pueden encontrar en la utilización de este software se puntualizan a continuación:

- La posibilidad de tener una vista 2D y 3D logra que el usuario pueda orientarse mejor y planee los recorridos. La vista 2D da un mapa del ambiente y la vista 3D posiciona al usuario dentro del ambiente.
- La utilización de sensores se vuelve parte esencial en el éxito de la conducción cuando el robot se acerca demasiado a un objeto considerado o no dentro del ambiente haciendo que el apoyo visual se sustituya por los datos del estado de los sensores.
- La conducción puede ser elegida por el usuario a través del joystick o del teclado que en la mayoría de las PC's están disponibles.
- La posibilidad de elegir entre una cuadrícula o una textura como piso para comodidad del usuario.
- La animación de la llanta delantera, para que el usuario tenga una mejor idea de la dirección en la que se moverá el robot.
- El reloj se agrega para que el usuario pueda medir el tiempo de los recorridos y con ello pueda determinar si es viable mantener al robot o no dentro del ambiente.
- El cuadro de ajuste permite hacer las modificaciones que adecuarán los avances y rotaciones dependiendo del tipo de motor que se tenga.
- La existencia de dos métodos para la detección de colisiones permite al sistema adecuarse a la exactitud que se requiera. El método exacto proporciona una mejor simulación de colisión.

- El usuario puede escoger el software de graficación que prefiera para la creación de objetos a insertar, siempre y cuando este software pueda guardar en formato .3ds, además estos archivos deberán ser convertidos al formato .X utilizando el convertidor proporcionado (Conv3ds.exe) u otro.

- Puede ejecutarse en un sistema que utilice un mínimo de hardware y Windows 9X, sin sistemas especializados que reduzcan su disponibilidad y aumenten su costo el cual fue parte del objetivo principal.



5. REFERENCIAS

[1] R. Osorio., J Silva., A. Arteaga., Desarrollo del software de un sistema para la operación de un Robot Móvil en un ambiente de Realidad Virtual 3D., Tesis, Facultad de Ingeniería UNAM., 2002.

[2] Constantino Sánchez Ballesteros. Ed. Alfaomega, "Programación multimedia avanzada con DirectX", 1992

[3] Casey Larijani,Ed. McGraw-Hill , "Realidad Virtual, L". Casey Larijani, Ed. McGraw-Hill, 1994, pág 31

[4] Date Becker ,Ed. Marcombo , "El gran Libro de 3D Studio Max". 1997

[5] Casey Larijani,Ed. McGraw-Hill , "Realidad Virtual, L". Casey Larijani, Ed. McGraw-Hill, 1994

[6] Stefan Gotts. Collision Queries using Oriented Bounding Boxes., Tesis de la Universidad de Carolina del Norte U.S. 2000. 192 p.

[7] Thomas Möller et. al.. Fast, minimum storage Ray/Triangle Intersection., Universidad Tecnológica de Chalmers.