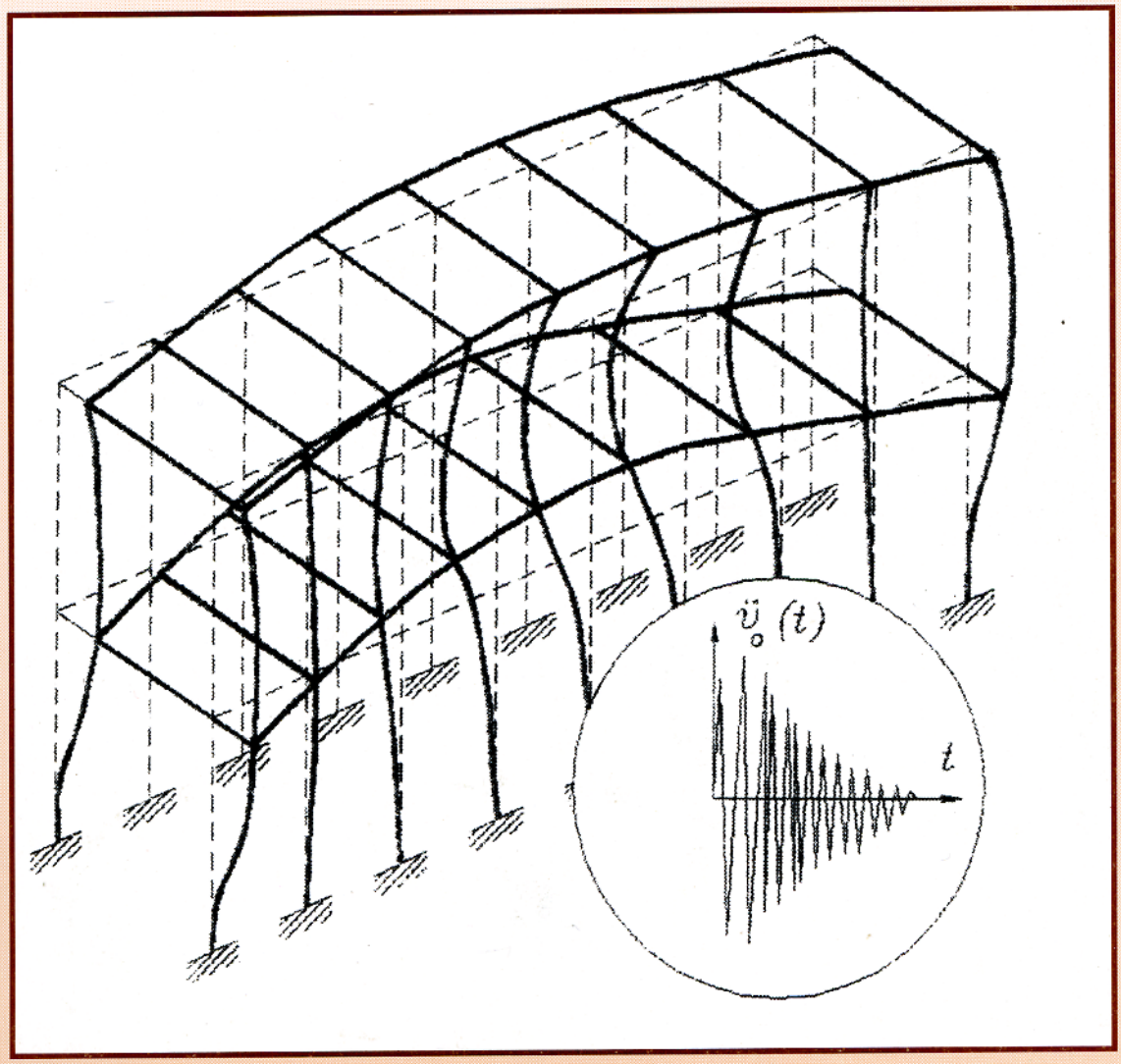
ISSN 1409-2441

Ingeniería

Revista de la Universidad de Costa Rica
Julio/Diciembre 1997 VOLUMEN 7 Nº 2



ISSN 1409-2441

INGENIERIA

Revista Semestral de la Universidad de Costa Rica
Volumen 7, Julio/Diciembre 1997 Número 2

DIRECTOR

Rodolfo Herrera J.

CONSEJO EDITORAL

Víctor Hugo Chacón P.
Ismael Mazón G.
Domingo Riggioni C.

CORRESPONDENCIA Y SUSCRIPCIONES

Editorial de la Universidad de Costa Rica
Apartado Postal 75
2060 Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

CANJES

Universidad de Costa Rica
Sistema de Bibliotecas, Documentación e Información
Unidad de Selección y Aquisiciones-CANJE
Ciudad Universitaria Rodrigo Facio
San José, Costa Rica

Suscripción anual:
Costa Rica: ¢ 1 000,00
Otros países: US $ 25,00

Número suelto:
Costa Rica: ¢ 750,00
Otros países: $ 15,00

Ingeniería 6 (2): 119-130, 1997 San José, Costa Rica

MECANISMOS DE INTERACCION EN MUNDOS VIRTUALES BASADOS EN V.R.M.L.

Carlos Vargas Castillo[*]
Victor Suárez Diaz[**]

Resumen

La Realidad Virtual se incorporó recientemente a la Red Internet a través del lenguaje estandar **V.R.M.L.**, el Lenguaje Modelador de Realidad Virtual. Se introducen los mecanismos de interación que permiten a los creadores de mundos virtuales desarrollar interfases altamente dinámicas usando **V.R.M.L.**

Summary

Virtual Reality has recentenly been incorporated to the Internet Network through the standard language **V.R.M.L.**, the Virtual Reality Modeling Language. Interactive mechanisms that allow to virtual world creators the developping of highly dynamic interfases are introduced.

1. INTRODUCCIÓN[1]

Existe concenso en que la realidad virtual es "la tecnología que brinda la experiencia de interactuar con un sistema basado en computador que representa un mundo virtual de escenas simuladas y sonidos"[2]. El mundo virtual es creado utilizando gráficos en tres dimensiones y elementos de audio. El mundo es generado en tiempo real por el computador; lo que implica que se puede navegar e interactuar con el mundo virtual y su contenido de acuerdo a las preferencias particulares del usuario. La imagen desplegada responde de acuerdo a sus acciones: adonde usted obseve, la dirección en que usted se mueve, o el objeto que usted manipule. El proceso de interacción se enriquece al utilizar equipo especializado, tal como cascos virtuales, visores estereoscópicos, o guantes alambrados. Sin embargo, aún con un simple ratón se puede navegar en un mundo virtual.

La construcción de un mundo virtual es útil para una gran variedad de aplicaciones, ya sean comerciales, industriales, científicas, médicas o para el entretenimiento[3]. A un mundo virtual se le controla mas fácilmente que al mundo real que representa. Para un arquitecto que necesite mostrar cómo las sombras caerán alrededor de su nuevo edificio, la realidad virtual puede hacer que el sol salga y se oculte tantas veces como lo ocupe.

El **V.R.M.L.**[4] *(Virtual Reality Modeling Language)* es un lenguaje tridimensional e interactivo orientado al modelaje y la visualización de objetos y mundos virtuales en red. Es un novedoso medio que permite construir y experimentar nuevos mundos

[*] Profesor Escuela de Ciencias de la Computación e Informática, Facultad de Ingeniería, UCR.
[**] Egresado Escuela de Ciencias de la Computación e Informática, Facultad de Ingeniería, UCR.

modelados con base a aspectos del mundo que conocemos o de mundos imaginados. El **V.R.M.L.** se distingue de otros lenguajes similares, en que fue diseñado específicamente para ser usado por aplicaciones que hagan acceso directo a Internet. Esto le da la gran ventaja de tener soporte en una amplia variedad de plataformas y le permite ser accesado por un vasto número de personas alrededor del mundo.

Este lenguaje nació como resultado de una conferencia sobre el *World Wide Web* llevada a cabo en 1994 en Ginebra, Suiza, en la cual se discutió acerca de interfases de realidad virtual para la Internet. En esta conferencia se contó con la participación de un connotado grupo de especialistas e investigadores sobre el tema, quienes propusieron la realización de una especificación la cual, meses más tarde, vino a culminar con la primera versión oficial del **V.R.M.L.**[5].

Esta primera versión se basó en una especificación ya existente desarrollada por la compañía *Silicon Graphics, Inc.*, denominada *Open Inventor*[6]. Esta especificación satisfacía varios requerimientos importantes, tales como independencia de plataforma, facilidades para desarrollar mundos de gran tamaño, e incluso la capacidad de trabajar en condiciones de baja amplitud de banda. Sin embargo, se evidenciaban fallas en varios aspectos. Lo más significativo: el énfasis fue hacia lo descriptivo, muy poco en lo interactivo y, absolutamente nada en lo multi-usuario.

Siendo el soporte a comportamientos interactivos una necesidad notoria, este se incorporó en la nueva especificación del **V.R.M.L.** El grupo de trabajo **VAG** *(VRML Architecture Group)* presentó en agosto de 1996 la primera especificación oficial, **(V.R.M.L.** version 2.0, estándar **ISO/IEC CD 14772)**[7], en ocasión de la exposición **SIGGRAPH'96** en Nueva Orleans. A la revisión mas reciente, liberada hace apenas pocos meses, se le denominó VRML97. Esta última especificación es una versión bastante madura con la característica que ofrece mecanismos para la definición de elementos que permiten incorporar interacción y dinamismo a los mundos virtuales modelados.

Cabe notar que existen herramientas autor que facilitan la tarea de generar mundos virtuales en VRML97. Sin embargo, usualmente no ofrecen soporte a la especificación completa del lenguaje. Algunas herramientas únicamente producen código para la versión 1.0 lo que obliga a utilizar un programa de conversión. En la práctica, lo mas común es que los elementos interactivos haya que incorporarlos directamente sin ayuda de herramientas de alto nivel. Debido a esta situación y dado que la programación de la interacción es uno de los aspectos mas complejos en la creación de mundos virtuales, los autores han volcado su experiencia en este trabajo que se centra exclusivamente en los mecanismos de interación en mundos virtuales basados en VRML97.

La capacidad de interacción dentro de los mundos virtuales le permite al usuario poder directamente influir sobre el comportamiento de los distintos objetos que se definen dentro del espacio virtual. Gracias a la interacción es posible simular la forma de actuar de objetos del mundo real, en donde por ejemplo, puede modelarse una lámpara de mesa que se encienda y apague cuando el usuario presiona su interruptor; las puertas de un establecimiento, que se abren automáticamente cuando el usuario se acerca a ellas o incluso, la presentación de publicidad animada conforme el usuario atravieza un centro comercial virtual.

2. LOS COMPONENTES ESENCIALES DEL VRML97

Independientemente de que un mundo virtual simule la Universidad de Costa Rica, el Museo Nacional o un edificio producto de su imaginación, el mundo se encuentra poblado de

objetos que se caracterizan por varios atributos. Geometría es la forma del objeto. Apariencia se refiere al tamaño, color, textura, iluminación y sombras del objeto que se le aplican a su geometría. Comportamiento es la reacción del objeto a eventos. Las leyes de la dinámica suelen controlar la interacción entre objetos, por ejemplo, colisión o fuerza gravitacional. El comportamiento también se refiere al sonido generado por un objeto. Un objeto podría representar un punto de vista que está ligado a un sensor. Un objeto podría ejecutar una tarea en función del tiempo transcurrido.

Los objetos se conectan entre sí usando jerarquías. Por ejemplo, un objeto complejo, tal como un automóvil, realmente está conformado por varios objetos individualmente construidos, tales como el volante, los pedales, la marcha, las puertas, y las llantas. Todos estos objetos deben ser conectados para crear un objeto dinámico que es capaz de abrir las puertas o arrancar en un momento predeterminado.

Cualquier objeto 3-D consiste de un conjunto de polígonos lógicamente conectados entre sí. Los polígonos son figuras 2-D con varios bordes. Una vez que los polígonos son ensamblados siguiendo la geometría del objeto, debe hacerse el acabado (*rendering*) del objeto que le da a éste una apariencia distintiva. El acabado representa el fenómeno óptico que ocurre cuando la luz interactúa con las superficies del objeto y se considera el color, las fuentes de luz, las texturas y la localización del observador.

El principal componente del VRML97 es un elemento llamado nodo, el cual permite describir tanto las figuras tridimensionales que son visibles en el mundo virtual, como sus propiedades tales como color, forma, iluminación, posición, orientación, etc. Además, con ellos pueden definirse elementos tales como relojes y sensores. Los nodos generalmente corresponden a un tipo específico, y contienen una serie de valores que definen sus atributos.

El siguiente fragmento de código corresponde al nodo *Cylinder* que permite definir las características de un cilindro, el cual en este caso tiene una altura de dos y un radio de una unidad gráfica. El VRML97 no utiliza un sistema de medida de longitud en particular, por lo que una unidad gráfica puede entenderse como un centímetro, un metro, un pie, o una yarda. Nótese además, que los campos del nodo se encuentran encerrados entre llaves.

```
Cylinder
{height 2.0 #altura del cilindro
radius 1.0 #radio del cilindro}
```

El VRML97 define una serie de tipos de datos predefinidos para los campos. Algunos de ellos son: lógicos (SFBool), de punto flotante (SFFloat/MFFloat), entero (SFInt32/MFInt32), para rotación (SFRotation/MFRotation), hileras (SFString/MFString), fecha y hora (SFTime), vectores de dos y tres dimensiones (SFVec2f/MFVec2f, SFVec3f/MFVec3f) y otros. Los campos cuyo tipo inicia con "SF" indican que aceptan un único valor, como color, un número, etc., mientras que los que inician con "MF" pueden aceptar una lista de valores.

Es posible hacer referencia a una instancia de un nodo en diferentes lugares del mundo virtual, de tal forma que sólo hace falta definirlo una única vez, y luego se puede referenciar desde cualquier ubicación. Por ejemplo, se puede definir una puerta y reutilizarla en varias secciones de una casa virtual. Para esto se usan los comandos "**DEF**", para definir el objeto, y "**USE**" para hacer referencia al mismo.

Para lograr la capacidad de interacción y dinamismo en los mundos virtuales, debe definirse un mecanismo de comunicación entre nodos basado en la emisión y recepción de eventos o mensajes (eventos de tipo *eventOut* y *eventIn*, respectivamente). Para definir la relación de comunicación entre dos nodos, es necesario especificar lo que se denomina una

ruta con el comando "**ROUTE TO**". La sintáxis es la siguiente:

```
ROUTE <NodoA.eventoDeSalida>
 TO <NodoB.eventoDeEntrada>
```

En donde se indica que cuando se produzca el evento "eventoDeSalida" en el nodo "NodoA", se envíe el mensaje "eventoDeEntrada" al nodo "NodoB", y de esta forma se establece la ruta entre el nodo "A" y el nodo "B". A continuación se muestra la sintáxis de aquellos nodos que permiten la interación[8].

Transform

children	#exposedField
translation	# exposedField
rotation	# exposedField
scale	# exposedField
scaleOrientation	# exposedField
bboxCenter	# field
bboxSize	# field
center	# exposedField
addChildren	# eventIn
removeChildren	# eventIn

TimeSensor

enabled	# exposedField
startTime	# exposedField
stopTime	# exposedField
cycleInterval	# exposedField
loop	# exposedField
isActive	# eventOut
time	# eventOut
cycleTime	# eventOut
fraction_changed	# eventOut

PositionInterpolator

key	# exposedField
keyValue	# exposedField
set_fraction	# eventIn
value_changed	# eventOut

OrientationInterpolator

key	# exposedField
keyValue	# exposedField
set_fraction	# eventIn
value_changed	# eventOut

TouchSensor

enabled	# exposedField
isActive	# eventOut
isOver	# eventOut
touchTime	# eventOut
hitPoint_changed	# eventOut
hitNormal_changed	# eventOut
hitTexCoord_changed	# eventOut

PlaneSensor

enabled	# exposedField
autoOffset	# exposedField
offset	# exposedField
maxPosition	# exposedField
minPosition	# exposedField
isActive	# eventOut
translation_changed	# eventOut
trackPoint_changed	# eventOut

CylinderSensor

enabled	# exposedField
diskAngle	# exposedField
autoOffset	# exposedField
offset	# exposedField
maxAngle	# exposedField
minAngle	# exposedField
isActive	# eventOut
rotation_changed	# eventOut
trackPoint_changed	# eventOut

SphereSensor

enabled	# exposedField
autoOffset	# exposedField
offset	# exposedField
isActive	# eventOut
rotation_changed	# eventOut
trackPoint_changed	# eventOut

VisibilitySensor

enabled	# exposedField
center	# exposedField
size	# exposedField
isActive	# eventOut
enterTime	# eventOut
exitTime	# eventOut

ProximitySensor

enabled	# exposedField
center	# exposedField
size	# exposedField
isActive	# eventOut
enterTime	# eventOut
exitTime	# eventOut
position_changed	# eventOut
orientation_changed	# eventOut

Collision

children	# exposedField
proxy	# field
collideTime	# eventOut
addChildren	# eventIn
removeChildren	# eventOut

Group	
children	# exposedField
bboxCenter	# field
bboxSize	# filed
addChildren	# eventIn
removeChildren	# eventOut

3. ELEMENTOS DINAMICOS

El VRML97 brinda la capacidad de animación de objetos en tiempo real dentro de mundos virtuales. Esto permite crear en el observador la sensación de dinamismo y movimiento, a la vez que se enriquece la escena gráfica del mundo virtual, ya que ahora no se compone únicamente de figuras tridimensionales estáticas, sino que éstas pueden simular el comportamiento de objetos del mundo real.

Por animación en el VRML97 debemos entender la capacidad de variar las características visibles del objeto, tales como su apariencia (color, textura, tamaño y/o forma) y su posición dentro del espacio virtual. Lo que llamamos animación debe diferenciarse de lo que comúnmente es sobreentendido: la presentación de un fragmento de vídeo, en donde lo que se hace es yuxtaponer una secuencia de cuadros a intervalos de tiempo regulares. La diferencia principal entre ambos tipos de animación, radica en que la técnica usada en VRML97 debe ser realizada en tiempo real y depende de la posición del observador, mientras que la otra corresponde a la presentación de una secuencia de imágenes estáticas y "prefabricadas" las cuales no admiten nigún tipo de interacción.

Con VRML97 se le puede dar vida a cualquiera de los objetos que se definen dentro de un mundo virtual. Por ejemplo, se puede tener una ciudad virtual en donde se hayan definido edificios, automóviles, y personas. Entonces mediante la aplicación de las técnicas de animación del VRML97, se consigue simular el movimiento de todos estos objetos, dando así un mayor realismo a los mundos virtuales.

La animación de los objetos se logra por medio de la aplicación de cambios o transformaciones a los objetos conforme transcurre el tiempo. Los cambios pueden corresponder a variaciones en la apariencia o en la posición de los objetos. En este documento nos enfocaremos en la animación de la posición de las figuras, lo que nos permitirá poder desplazarlas a través del mundo virtual.

Para lograr el movimiento de los objetos, utilizamos el nodo *Transform*, el cual permite trasladar, rotar o variar la escala de un sistema de coordenadas asociado a uno o varios objetos. Si se varía el orígen de un sistema de coordenadas, todos los objetos asociados al mismo serán desplazados en igual magnitud y dirección.

Para poder controlar el momento en el que se realizan las transformaciones para la animación, podemos apoyarnos en el nodo *TimeSensor*, el cual genera eventos conforme transcurre el tiempo. El reloj definido por este nodo puede regularse para que genere los eventos cada segundo, milisegundo, etc., de acuerdo a las necesidades propias de la animación.

Contando con la capacidad de aplicar transformaciones a los objetos mediante el nodo *Transform* y la de disponer de un controlador de tiempo como el *TimeSensor*, únicamente hace falta coordinar ambas capacidades para lograr la animación de objetos tridimensionales. Estos conectores entre el nodo *TimeSensor* y el nodo *Transform* se conocen como nodos interpoladores. Se dispone de varios tipos, entre ellos *PositionInterpolator* y *OrientationInterpolator*.

Los interpoladores permiten asociar un momento en el tiempo (véase el campo *key* en la sintáxis de los nodos interpoladores), con una posición u orientación en el espacio tridimensional (véase campo *keyValue*). De este modo, se divide un intervalo de tiempo en un número dado de segmentos no necesariamente de igual magnitud y, a cada uno de ellos, se les

asocia una transformación, ya sea posicional o de orientación.

Para aclarar el concepto anterior, tómese por ejemplo el intervalo de un minuto. Podemos dividir el minuto en cuatro segmentos separados entre sí por 15 segundos y etiquetarlos con las letras a, b, c, y d, como se muestra en la figura No. 1. Además, definamos las cuatro posiciones coordenadas que ocupará el objeto como c1, c2, c3 y c4, entonces se puede asociar cada "instante" con una coordenada tridimensional de la siguiente manera:

Instante	Coordenada
a	c1
b	c2
c	c3
d	c4

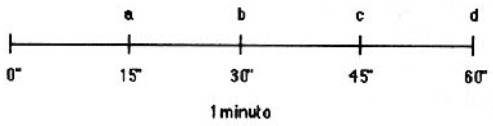


Figura No. 1. Representación de un minuto dividido en 4 segmentos de 15 segundos.

Esto significa que el objeto por animar deberá trasladarse hacia la posición "c1" cuando el reloj indique que han transcurrido 15 segundos (instante "a"), luego cuando se haya llegado a los 30 segundos (instante "b"), el objeto deberá trasladase hacia la coordenada "c2" y así, sucesivamente, hasta que haya transcurrido el minuto, momento en el cual el objeto se encontrará en la coordenada "c4" y habrá concluido la animación.

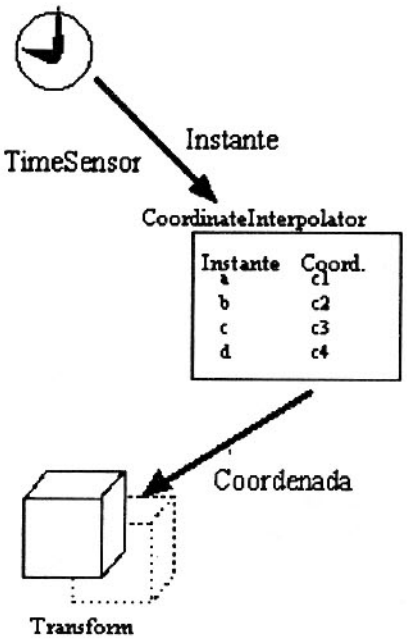


Figura No. 2. Generación de eventos para indicar una nueva ubicación para el objeto.

La relación que se establece entre el nodo *TimeSensor* y el nodo *Transform* a través de los nodos interporladores, se da gracias a la capacidad del VRML97 para el manejo de eventos (obsérvese la figura No. 2). Esto permite que el nodo *TimeSensor* genere un evento que es recibido e interpretado por uno de los nodos interpoladores, el cual al recibirlo resuelve la relación "instante-transformación" (o *key-keyValue*), y envía a su vez un mensaje al nodo *Transform* indicándole el tipo y condiciones de la transformación, para que éste modifique el sistema de coordenadas del objeto asociado a éste y se logre así la animación del mismo.

4. PERCEPCION DE LAS ACCIONES REALIZADAS POR EL USUARIO

Para posibilitar la interacción entre el usuario y el mundo virtual se utilizan una serie de nodos de tipo sensor, que fueron diseñados para percibir ciertas acciones realizadas por el

usuario y para generar señales que luego sean interpretadas por los objetos asociados para iniciar animaciones.

Conviene distinguir los dos grupos de sensores usados para percibir las acciones del usuario. Están los sensores que permiten percibir acciones hechas con el cursor (puntero del ratón o cualquier otro dispositivo de entrada), tales como presionar un botón,, arrastrar un objeto o, sencillamente, mover el cursor sobre un objeto. También se dispone de otro grupo de sensores que perciben el movimiento del avatar conforme éste se desplaza a través del mundo virtual. Un avatar es una representación del usuario dentro del mundo virtual, la cual tiene cierto volúmen e incluso puede especificársele forma y apariencia. Los sensores para detectar acciones del cursor son: *TouchSensor*, *PlaneSensor, CylinderSensor* y *SphereSensor*. El nodo *TouchSensor* permite detectar el momento en el que se mueve o se presiona el botón sobre un objeto. Los nodos "*PlaneSensor*", *CylinderSensor* y *SphereSensor* al ser asociados a un objeto, le permiten al usuario desplazarlo a través del mundo virtual. La dirección y forma del desplazamiento dependerá del tipo de sensor utilizado.

En el caso de *PlaneSensor*, éste permite desplazar los objetos sobre un plano imaginario. Por ejemplo, si se modela una silla y se asocia a un nodo *PlaneSensor* definido sobre el suelo del mundo virtual, el usuario podrá mover la silla sobre el piso y en cualquier dirección, siempre y cuando ésta no se salga del área definida para el plano. Obsérvese la figura No. 3.

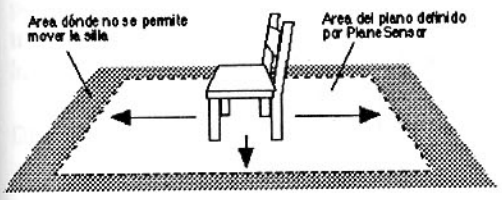


Figura No. 3. La silla puede moverse en cualquier dirección sobre el suelo, dentro del plano definido por el nodo PlaneSensor.

Los objetos asociados a un nodo *CylinderSensor* pueden rotarse con respecto al eje "Y" del sistema coordenado definido para el objeto. Por ejemplo, se puede asociar un nodo *CylinderSensor* con el volante de un automóvil. El sistema coordenado es definido de tal forma que su eje "Y" atraviese perpendicularmente su centro, como se observa en la figura No. 4.

Si se desea una mayor flexibilidad para la rotación, puede emplearse el nodo *SphereSensor*, con el cual puede hacerse girar el objeto asociado en cualquier dirección. Lo que se persigue es crear la sensación de estar haciendo girar una esfera y, conforme ésta gira, el objeto asociado a ésta gira proporcionalmente en la misma dirección.

Los nodos sensores para detectar el desplazamiento del avatar a través del mundo virtual son: *VisibilitySensor*, *ProximitySensor* y *Collision*. El nodo *VisibilitySensor* está diseñado para detectar si un objeto se encuentra dentro del rango visual del usuario, considerando su posición y orientación. De esta forma, en el momento en el que el objeto entra dentro del campo de visión del usuario, el nodo *VisibilitySensor* genera un evento que es enviado al objeto asociado para que éste inicie la animación respectiva. De igual forma sucede cuando el objeto sale de la visibilidad del usuario, generándose otro evento que lo indica. Con ello es posible, por ejemplo, tener un rótulo luminoso que empiece a parpadear sólo cuando el usuario voltea a verlo, economizando así el trabajo del visualizador.

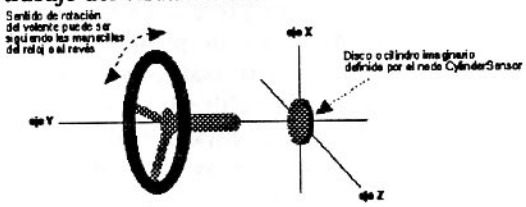


Figura No.4. Relación entre el disco imaginario definido por el nodo *CylinderSensor* ubicado en el centro del sistema coordenado con el volante del automóvil.

El nodo *ProximitySensor*, se asemeja bastante al sensor anterior y permite detectar el momento en el que el avatar entra o sale de una región asociada a un objeto. Esta región corresponde al volúmen de una caja imaginaria con centro y tamaño definidos en el nodo *ProximitySensor*, de tal forma que cuando el usuario ingresa dentro de esta caja o sale de ella, se generan eventos que pueden activar o detener una animación, según se requiera.

Finalmente tenemos el nodo sensor *Collision* que permite detectar el momento en el que el avatar toca un objeto dentro del mundo virtual, como por ejemplo una pared o una mesa. Con ello es posible producir acciones que retroalimenten al usuario sobre su posición. Se puede especificar por ejemplo la generación de un sonido fuerte cuando se colisiona con una pared, o con un objeto caliente.

5. EJEMPLO DE UN MUNDO VIRTUAL INTERACTIVO

Observemos ahora como se aplican los conceptos vistos en las secciones anteriores con un ejemplo en el que crearemos una escena virtual con capacidades interactivas. La escena es relativamente simple y lo que se persigue es activar una animación al hacer presionar el botón del ratón sobre uno de los objetos modelados, llamado "objeto sensible".

Empezaremos por definir dos objetos diferentes formados por un grupo de figuras básicas (cilindros y esferas) que representen a una nave espacial (figura No. 5) y a un platillo volador (figura No. 6). La nave espacial, está formada por dos esferas y un disco cilíndrico, los cuales se han achatado para darles una forma aerodinámica. El platillo volador es mucho más simple y lo forman una esfera y un disco cilíndrico:

```
# Nave espacial
DEF NaveEstacionaria Group
children
    # Alas
    DEF AlaNave Transform
        scale 0.5 1.0 1.5  # achatamiento
        children Shape
        appearance DEF Blanco Appearance
            material Material
            geometry Cylinder
            radius 1.0
            height 0.025

    # Fuselaje
    DEF FuselajeNave Transform
        scale 2.0 0.2 0.5  # achatamiento
        children Shape
            appearance USE Blanco
            geometry Sphere

    # Detalle del ala y la cabina
    Transform
        scale 0.3 2.0 0.75# achatamiento
        children
            USE AlaNave,
            USE FuselajeNave
```

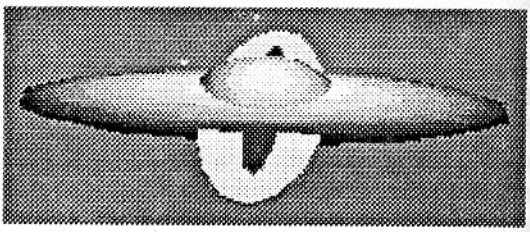


Figura No. 5. La nave espacial vista desde un costado.

```
# Platillo Giratorio
DEF OvniGiratorio Transform
children

Shape # Núcleo del Ovni
        appearance Appearance
        material Material
        geometry Sphere

Shape # El plato
        appearance Appearance
        material Material
        geometry Cylinder
```

```
radius 2.0
height 0.05
```

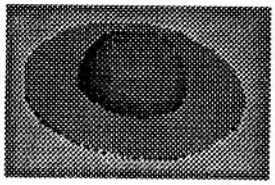


Figura No. 6. Platillo volador formado únicamente por una esfera y un cilindro en forma de disco.

Habiendo definido estos objetos mediante el comando "**DEF**", podremos hacer luego referencia a ellos para desplegarlos con el comando "**USE**", sin necesidad de volver a reescribir de nuevo el código.

Ahora necesitaremos definir como se comportará la animación de uno de los dos objetos. Para ello, dejaremos a la nave estática, y haremos que el platillo gire alrededor de ella formando un círculo completo. Esto implica en primer lugar, que tendremos que ubicar la nave en el centro del sistema coordenado del mundo virtual, y el platillo a cierta distancia de ella (cinco unidades).

Esto se logra insertando una traslación inmediatamente después de abrir las llaves del nodo *Transform*: "translation 5.0 0.0 0.0" lo cual indica que se desplace el centro del platillo 5 unidades sobre el eje X. Luego para lograr que gire con respecto al centro de la nave, debemos volver a colocar el centro del sistema coordenado relativo al platillo, en donde estaba, por lo que debemos insertar luego de la traslación: "center -5.0 0.0 0.0". Obteniéndose lo siguiente:

```
DEF OvniGiratorio Transform
   # desplazamiento
   translation 5.0 0.0 0.0
   # reubicación del centro
   center -5.0 0.0 0.0
```

Para guiar la animación requeriremos de un reloj controlador de tiempo y al que le definiremos un intervalo de 3.5 segundos:

```
DEF RelojDeAnimacion TimeSensor
   { cycleInterval 3.5 }
```

La ruta de animación la definiremos para tres instantes distintos: inicial 0.0, medio 0.5 y final 1.0, a los cuales debemos asociarles las tres transformaciones a ser aplicadas sobre el platillo, que en este caso corresponden a rotaciones alrededor del centro de la nave espacial. De esta forma podemos definir la ruta de animación así:

```
DEF RutaDelOvni OrientationInterpolator {
   key [ 0.0, 0.5, 1.0 ]  # instantes de tiempo
   keyValue [
      0.0 1.0 0.0  0.0,      #                1er
transformación
      0.0 1.0 0.0  3.14, # 2da transformación
      0.0 1.0 0.0  6.28      #                3ra
transformación
```

La anterior definición asocia cada uno de los tres diferentes instantes de tiempo con una transformación. El primer instante (0.0) se asocia con la rotación de 0.0 radianes, el segundo instante (0.5) con una rotación de 3.14 radianes, y el tercer instante (1.0) con una rotación de 6.28 radianes. Los primeros tres valores de cada rotación corresponden a una coordenada tridimensional (x y z), que junto con las coordenadas del origen, definen un vector con respecto al cual se realizará la rotación. De esta forma, las coordendas que definimos especifican un vector paralelo al eje "Y" y que hará que el platillo se desplace girando por los costados de la nave.

Necesitamos hacer que el platillo pueda percibir al usuario presionando el boton del ratón. Esto lo logramos insertando la definición del nodo *TouchSensor* en la especificación del platillo al mismo nivel en que se definen las figuras del

mismo (como componente del campo *children*), para que todas ellas puedan ser sensibles:

```
DEF OvniGiratorio Transform
children

DEF SensorDeTacto TouchSensor
```

Finalmente, se requiere unir las diferentes piezas que ligan el evento generado por el usuario, con la activación del reloj que a su vez enviará mensajes al platillo para que este se desplace. Lo anterior debe hacerse por medio del comando "**ROUTE TO**":

```
# Ruta entre el sensor de tacto y el reloj
   ROUTE SensorDeTacto.touchTime
   TO RelojDeAnimacion.set_startTime
# Ruta entre el reloj y la ruta de animacion
   ROUTE
RelojDeAnimacion.fraction_changed
   TO RutaDelOvni.set_fraction
# Ruta entre la ruta de animacion y el Ovni
   ROUTE RutaDelOvni.value_changed
   TO OvniGiratorio.set_rotation
```

Disponiendo ahora de todos los elementos necesarios, podemos ensamblar un mundo virtual interactivo con el programa mostrado a continuación, en donde, por razones de espacio, fue necesario suprimir algunas partes, pero que pueden ser tomadas de los segmentos de programa anteriores. La figura No. 7 muestra una secuencia de imágenes de la animación.

```
#VRML V2.0 utf8
Group
   children
   # Nave espacial estacionaria
   DEF NaveEstacionaria Group

   # Ovni giratorio
   DEF OvniGiratorio Transform

   # Reloj de animacion
   DEF RelojDeAnimacion TimeSensor
   cycleInterval 3.5

# Ruta de animacion para el ovni giratorio
DEF RutaDelOvni
OrientationInterpolator

ROUTE SensorDeTacto.touchTime
TO RelojDeAnimacion.set_startTime

ROUTE
Reloj DeAnimacion.fraction_changed
TO RutaDelOvni.set_fraction

ROUTE RutaDelOvni.value_changed
TO OvniGiratorio.set_rotation
```

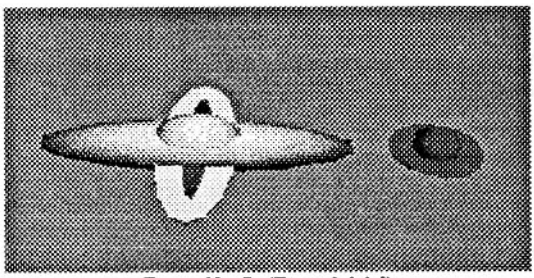


Figura No. 7a **(Etapa inicial)**

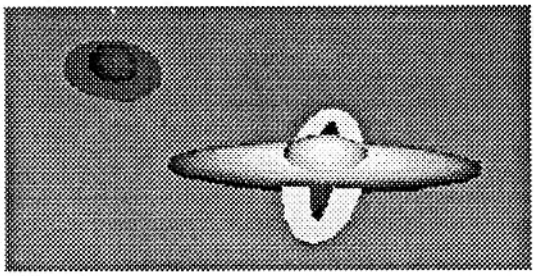


Figura No.7b.

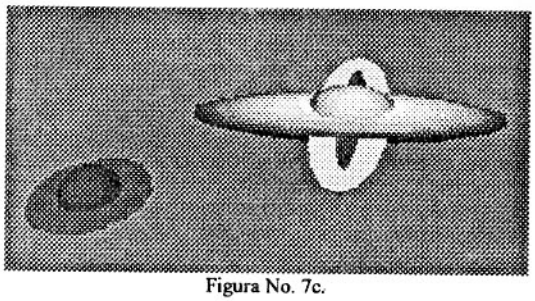


Figura No. 7c.

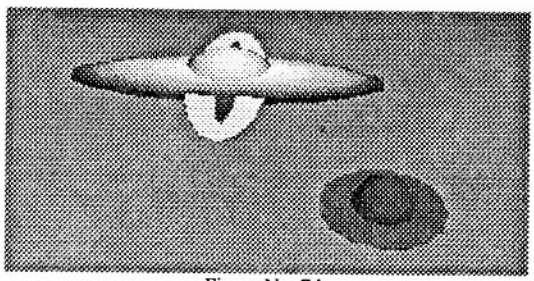


Figura No. 7d.

Figura No.7. **Secuencia de imágenes que muestra varias etapas de la animación realizada, luego de que el usuario presiona el puntero sobre el platillo.**

6. CONCLUSIONES

El desarrollador de aplicaciones de realidad virtual en la Internet debe tener muy presente que el potencial del VRML97 para crear mundos interactivos no está siendo totalmente explotado por los modeladores. Por ello, al momento presente, si se desea obtener un máximo uso de las capacidades ofrecidas, deberá utilizarse directamente el lenguaje VRML97, en el tanto que no aparezcan versiones actualizadas de los modeladores.

Hemos intentado transferir la experiencia adquirida en este novedoso campo de la manera mas comprensible. Sin embargo, una apreciación mas clara y completa se logra únicamente con la práctica, particularmente en lo que respecta al manejo de las capacidades dinámicas e interactivas del VRML97. Efectos sonoros y figuras sensibles que invoquen la carga de otro mundo virtual son algunas de las características interesantes que podrían agregarse al ejemplo mostrado en la última sección.

6. REFERENCIAS

(1) Este artículo se desarrolló dentro del proyecto de Investigación 326-97314 "Desarrollo y Utilización de la Realidad Virtual en la Internet Gráfica", dirigido por Carlos Vargas Castillo.

(2) Una definición mas amplia en [Jacobson, L., 1994] y en [Pesce, M., 1996].

(3) Todo el Número *de IEEE Computer* está dedicado a mostrar aplicaciones diversas de realidad Virtual.

(4)Las direcciones http://www.sdsc.edu/VRML y http://www.hike.te.chiba.ac.jp/ikeda/documentation/VRML/whatis.html son repositorios de información general sobre el VRML Incluyen visualizadores, modeladores, herramientas, documentación y muestras de mundos virtuales.

(5) La especificación de la primera versión del VRML se encuentra en la dirección http://vag.VRML.org/VRML10c.html.

(6) Abundante y actualizada información sobre el lenguaje *Open Inventor* en la direccción http://www.sgi.com/Technology/Inventor.html.

(7) La especificación de la segunda versión del VRML se encuentra en las direcciones http://vag.VRML.org/VRML10c.html y en http://VRML.sgi.com/moving.worlds/spec/index.html.

(8) Un tratamiento amplio y práctico sobre la temática en [Ames, A.,1996].

7. BIBLIOGRAFÍA

Ames, Andrea L. et. al. "*VRML 2.0 Sourcebook*", John Willey & Sons Inc., New York, 1996

Krüger, W. et. al. *The Responsive Workbench: A Virtual Work Enviroment*. "IEEE Computer: Virtual Environments", Vol. 28, No. 7, Julio 1995.

Jacobson, Linda. "*Garage Virtual Reality*", Sams Publishing, Illinois, 1994.

Pesce, Mark. VRML para Internet, New Riders, México, 1996.