X3d2pov. Traductor de X3D a POV-Ray

X3d2pov. Traductor of X3D to POV-Ray

Andrea Castellanos Mendoza,[1] Diego Alfonso Ahogado,[2] Jean Pierre Charalambos H.[3]

RESUMEN

El problema de representar objetos tridimensionales en medios digitales es un campo de estudio en computación gráfica que presenta diversos enfoques de solución. Hoy en día existen herramientas que permiten llevar a cabo esta representación de objetos, teniendo en cuenta el tipo de aplicación que se requiera. X3D (extensible 3D) es un estándar extensible que puede ser soportado fácilmente por *Web Browsers* y que está diseñado para ser de alto rendimiento, con el fin de conseguir una interacción eficiente del usuario con el modelo en tiempo real. El trazador de rayos POV-Ray (Persistence of Vision Raytracer), por el contrario, genera imágenes tridimensionales fotorrealistas de alta calidad con un bajo rendimiento. En este artículo se expone el diseño de una solución en desarrollo que pretende facilitar la traducción del código XML utilizado para la representación de objetos descritos mediante el estándar X3D a código interpretable por el trazador de rayos POV-Ray, con el fin de conseguir generar dichos objetos con una alta calidad.

Palabras clave: computación gráfica, formatos 3D, POV Ray, X3D, análisis sintáctico, generación de imágenes.

ABSTRACT

High-quality and low-quality interactive graphics represent two different approaches to computer graphics' 3D object representation. The former is mainly used to produce high computational cost movie animation. The latter is used for producing interactive scenes as part of virtual reality environments. Many file format specifications have appeared to satisfy underlying model needs; POV-ray (persistence of vision) is an open source specification for rendering photorealistic images with the ray tracer algorithm and X3D (extendable 3D) as the VRML successor standard for producing web virtual-reality environments written in XML. X3d2pov has been introduced to render high-quality images from an X3D scene specification; it is a grammar translator tool from X3D code to POV-ray code.

Keywords: computer graphics, 3D standard formats, POV-ray, X3D, parsing, image rendering.

Recibido: febrero 27 de 2006
Aceptado: junio 23 de 2006

Introducción

Dentro de la gran variedad de lenguajes de modelamiento de objetos tridimensionales, POV-Ray y X3D se destacan por su gran aceptación en sus diferentes campos de aplicación. POV-Ray es un lenguaje de programación escrito por David K. Buck y Aaron A. Collins (POV-Team, 1999) y en la actualidad se considera el estándar de uso libre más conocido que utiliza el trazado de rayos como técnica para generar imágenes con un alto grado de realismo (Wald y Slusallek, 2001). Este lenguaje contiene librerías, figuras y texturas predefinidas, sobre las cuales se describen escenas en un archivo de texto. Las imágenes se generan mediante un algoritmo de trazado de rayos (propio de POV-Ray) a partir de la definición de los objetos que conforman la escena y la cámara desde la cual se va a visualizar. Aunque el proceso de interpretación de las escenas y generación de su respectiva imagen no es un procedimiento rápido (por la técnica propia del trazado de rayos), se suelen obtener imágenes con efectos de reflexión, sombras y transparencias de muy alta calidad.

[1] Ingeniera de sistemas. M.Sc. en Telecomunicaciones, Universidad Nacional de Colombia, Bogotá. Gerente de Proyectos, Platika Ltda. Instructor de Redes Cisco, Universidad Nacional de Colombia, Bogotá. edyandrea@yahoo.es

[2] Ingeniero de sistemas. M.Sc. en Telecomunicaciones, Universidad Nacional de Colombia, Bogotá. Technische Universität Darmstadt, Darmstadt, Alemania: International Master in Information and Communication Engineering, segundo semestre en curso. Investigador, grupo GITUN (Grupo de investigación en Telecomunicaciones, Universidad Nacional de Colombia, Bogotá. daahogadoa@unal.edu.co

[3] Ingeniero industrial.M.Sc. en Ingeniería de Sistemas. Estudiante de doctorado en *software*, Universidad Politécnica de Calatuña, España.Profesor asistente, Departamento de Ingeniería de Sistemas e Industrial, Universidad Nacional de Colombia, Bogotá. jpcharambosh@unal.edu.co, charalam@lsi.upc.edu.

Por otra parte, X3D es un estándar que ha sido proclamado por el "Web3d Consortium" como el futuro sucesor de VRML, que ha venido siendo empleado ampliamente para la generación de mundos virtuales. Aunque aún se encuentra en su fase de especificación, gracias a una arquitectura basada en componentes, el nuevo estándar presenta un mecanismo efectivo de extensibilidad, sin perder ninguna de las capacidades presentes en su predecesor. Adicionalmente, debido a que X3D utiliza la sintaxis de XML,[4] no es complicado entender la disposición de la información en el archivo, y además es posible apoyarse en un analizador sintáctico de XML para extraer y manejar la información descrita en el estándar X3D. Este lenguaje ha sido creado con el propósito de especificar escenas 3D dinámicas a través de las cuales se puede navegar e interactuar, empleando un *web browser plug-in,* o en algunos casos, una aplicación *stand-alone* (independiente). En cualquiera de los dos eventos, la generación de los gráficos está basada en librerías como OpenGL, Direct3D u otro sistema acelerado en *hardware*,[5]. Típicamente en una escena descrita en X3D el usuario posee la capacidad de navegar por el espacio 3D e interactuar con algunos de sus objetos constituyentes, lo que supone la necesidad de dibujar continuamente el entorno. Dada la capacidad de cómputo actual, en un escenario promedio este proceso se puede realizar sin pérdida de continuidad únicamente empleando algoritmos eficientes de dibujo (normalmente basados en librerías soportadas por las tarjetas graficadoras), y no mediante aquellos concebidos para sintetizar imágenes con un alto nivel de detalle.

En este artículo se presenta X3D2POV, una herramienta de *software* en desarrollo creada para permitir la traducción de los modelos descritos bajo X3D a *scripts* POV-Ray, y proporcionar de esta forma un trazador de rayos para objetos definidos en lenguaje X3D. En la segunda y tercera secciones se revisa la especificación de objetos tridimensionales básicos en X3D y POV-Ray, respectivamente; en la cuarta, se presentan las herramientas usadas para realizar el análisis sintáctico y la traducción de formatos; finalmente, en la quinta sección se presentan los resultados obtenidos y las expectativas futuras de la aplicación.

Especificación de objetos en X3D

El Web3D Consortium, perteneciente al W3C (*World Wide Web Consortium*), ha considerado el crecimiento y reconocimiento de XML y definido X3D (Extensible 3D), un estándar 3D compatible con las funcionalidades de VRML pero que amplía sus capacidades y que brinda, además, un lenguaje jerárquico y extensible (WEB 3D Consortium, 2003a). El estándar de representación tridimensional X3D aprovecha la funcionalidad de XML para la especificación de su sintaxis dado que éste último brinda una estructura de etiquetas con un estilo similar al utilizado en los documentos HTML. Por otra parte, esta sintaxis brinda mayor naturalidad en el lenguaje que la existente en VRML97, y permite su integración con páginas *Web* bajo un formato simple y de próxima generación (Holmes, 2002).

Primitivas básicas

Las primitivas básicas se definen dentro de la jerarquía de figuras, es decir, se ubican dentro de las etiquetas `<Shape></Shape>`. La Tabla 1 muestra las primitivas básicas de X3D y sus valores por defecto (*web* 3D Consortium, 2003b; Brutzman y Muñoz, 2003).

Tabla 1. Primitivas básicas

Figura	*Etiqueta*	*Atributos*	*Valor por Omisión*
Caja	`<Box>`	size	`size = '1 1 1'`
Cono	`<Cone>`	bottomRadius (Radio de la base del cono) height (altura del cono).	`bottomRadius = '1' height = '2'`
Cilindro	`<Cylinder>`	height (altura), radius (radio del cilindro), bottom y top (Booleanos que representan la existencia o ausencia de la cara inferior y superior del cilindro, respectivamente)	`height = '2' radius = '1' bottom = 'true' top = 'true'`
Esfera	`<Sphere>`	radius (radio de la esfera)	`Radius= '1'`
Texto	`<Text>`	string, length como atributos obligatorios.	

Apariencia

La etiqueta `<Material>` se emplea para definir las características de la superficie del objeto definido dentro de la misma etiqueta *Shape* en donde este se encuentra. Se maneja en la forma que los elementos tridimensionales, declarándolos y modificando los atributos que no se desee que sean tomados por defecto. En la Tabla 2 se presentan los atributos que se pueden definir, con su respectivo valor por defecto.

Transformaciones geométricas

Adicionalmente, el estándar X3D permite definir transformaciones espaciales para las figuras situadas en el es-

[4] XML (*Extensible Markup Language*) es un estándar para la definición de lenguajes basados en etiquetas, utilizados para el almacenamiento de información, de manera jerárquica y similar al estándar HTML.

[5] OpenGL y Direct3D son librerías desarrolladas para programar aplicaciones que utilizan el potencial gráfico de los computadores y facilitan la manipulación de objetos tridimensionales.

cenario, utilizadas principalmente para realizar traslaciones, rotaciones y escalamientos de cada figura. Estas transformaciones se encontrarán en nodos de tipo <transform>, que se utilizan como padres de los nodos <shape> (figura) para alterar el estado de una o más figuras que se encuentren en un nivel jerárquico inferior. Los tipos de transformaciones se utilizan dando valores a los atributos de <transform>, de los cuales los comúnmente utilizados se muestran en la Tabla 3.

Tabla 2. Atributos de la etiqueta <Material>

Atributo	*Utilización*	*Valor por omisión*
diffuseColor	Define el color RGB normalizado que deberá tener la superficie del elemento 3D.	'1 1 1'(Blanco)
shininess	Define el nivel de brillo de la superficie.	'0.2'
specularColor	Define el nivel de reflejo de luz de la superficie.	'0 0 0'
emmisiveColor	Cuánta luz brillante es emitida desde el objeto.	'0 0 0'
ambientIntensity	Cuánta luz de ambiente en todas las direcciones es reflejada desde todas las fuentes de luz.	'0.2'
transparency	Nivel de transparencia, 1: transparente, 0: completamente opaco.	'0'

Tabla 3. Transformaciones en X3D

Atributo	*Valores*	*Valor por defecto*
translation	Coordenadas de posición 'x y z'	'0 0 0'
rotation	Eje de rotación, ángulo (radianes)	'0 0 1 0'
scale	Escala por eje de coordenadas x' y z'	'1 1 1'

Cámaras y fuentes de iluminación

Aunque la iluminación y posición de cámara (o punto de vista) es establecida por defecto en la generación de las imágenes, X3D permite definir puntos de vista alternativos y fuentes de luz adicionales. La definición de estos elementos, se realiza mediante la adición de nodos <Viewpoint> y <PointLight>, ver Tabla 4.

Tabla 4: Puntos de Vista y Fuentes de Luz

Etiqueta	*Atributo(s)*	*Valor por defecto*
<Viewpoint>	position: Define la posición de la cámara 'x y z'.	'0 0 10'
<PointLight>	location: Define la posición de la luz. color: Define el color de la luz puntual	Location: '0,0,0' Color: '1,1,1'

Especificación de objetos en POV-Ray

Los *scripts* del trazador de rayos POV-Ray (POV TEAM. POV-Ray 3.5 Users Documentation, 2003), tienen una estructura muy diferente a la que es propia de XML, puesto que se asemejan más a las definiciones de estructuras de datos en lenguajes de programación como C. El objetivo del desarrolló de la traducción se refiere precisamente a la generación de código interpretable por el trazador de rayos que se base en modelos definidos en el estándar X3D mediante XML. Para alcanzar este objetivo, lo primero es realizar un estudio de la estructura de las definiciones de objetos en POV-Ray.

Primitivas básicas (Costas *et al.*, 2003)

Todas las figuras en POV-Ray se definen como una estructura de datos, en donde se indican vectores de coordenadas encerrados por los caracteres "<" y ">" para indicar las coordenadas espaciales necesarias para representar cada figura. Además, en los *scripts* para POV-Ray las figuras siempre deben llevar declarado explícitamente el atributo de pigmento (o un atributo de textura) que se hace mediante la declaración del atributo *pigment*, que indica el color, RGB o predefinido, que deba poseer la superficie del objeto. A continuación se hace una lista de las primitivas en POV-Ray y su estructura correspondiente:

Tabla 5. Principales primitivas de POV-Ray

Figura	*Ejemplo*

Caja	box {
	/* coordenadas requeridas */
	<-1,0,-1>, //Esquina delantera inferior izquierda
	<1,0.5,3> //Esquina trasera superior derecha
	pigment {color rgb <0,0,0>}
	}

Cono	Cone {
	/*coordenadas requeridas*/
	<0,1,0>, 0.3 //Centro y radio de una base
	<1,2,3>, 1.0 //Centro y radio de la otra base
	pigment {color rgb <0,0,0>}
	}

Cilindro	cylinder {
	/*coordenadas requeridas*/
	<0,1,0>, // Centro de una base
	<1,2,3>, // Centro de la otra base
	0.5 // Radio
	open // No dibuja las bases
	pigment { color rgb <0,0,0> }
	}

Esfera	sphere {
	/*coordenadas requeridas*/
	<.5,0,0>, .8 //Centro y radio
	pigment { Yellow }
	}

Apariencia

El atributo color se puede definir en términos de la combinación de canales RGB, o incluyendo una librería de colores predefinidos. Dos de los acabados básicos que se manejan, comparables a los atributos de material utilizados en X3D,

son *phong* y *specular.*[6] Estos dos atributos, como parte del acabado, deben pertenecer a una estructura de acabados (*finish*), que se debe declarar dentro del objeto, de esta manera:

```
sphere { <0,0,0>, 1
    pigment { White }
    finish {
     phong .25
     specular 1
    }
}
```

Transformaciones geométricas

Las modificaciones geométricas básicas, el escalamiento, la traslación y la rotación, se consideran como atributos de los objetos sobre los que se aplican. Como la ilustración de lo anterior considere el siguiente código:

```
sphere { <0,0,0>, 1
    pigment { White }
    scale 0.75
    rotate <-20,25,0>
    translate y
}
```

Cámaras y fuentes de iluminación

A diferencia del estándar X3D, POV-Ray exige la definición de una cámara que representa el punto de vista del objeto tridimensional. Esto se debe a que las escenas que se desean generar son estáticas. De otra parte, una fuente de luz es aquella que produce la iluminación del escenario para permitir que el objeto sea visible a la cámara. Con el fin de no limitar la visibilidad de los detalles y características de un objeto, en una escena se pueden utilizar varias fuentes de iluminación. Tanto la cámara como la fuente de iluminación se definen (al igual que el resto de objetos de POV-Ray), como una estructura de datos que contiene atributos en su interior.

Para el caso de las cámaras, se manejarán dos atributos: posición (*location*) y punto de enfoque (*look_at*). Un ejemplo de definición de una cámara es el siguiente:

```
camera {
    location <0, 2, -3>
    look_at <0, 1, 2>
    }
```

En el caso de las fuentes de luz puntuales no tiene sentido definir un punto de enfoque, puesto que la luz se dispersa en todas las direcciones mientras no encuentre obstáculo alguno. De esta manera, tenemos la siguiente estructura para la definición de fuentes de luz puntuales:

```
light_source {<2, 4, -3> color White}
```

Conversión entre especificaciones

Existen herramientas genéricas para efectuar el análisis sintáctico de un libreto de programación cualquiera respecto de una gramática dada. Sin embargo, gracias a su misma concepción, el análisis sintáctico de un archivo descrito mediante XML se realiza de un modo bastante más simple. Como X3D se encuentra descrito mediante XML, antes de describir el algoritmo empleado para la conversión a POV-Ray, primero se presentará el XML y se justificará la herramienta seleccionada para realizar su análisis sintáctico.

Análisis sintáctico mediante XML

XML es un conjunto de especificaciones que conforman el estándar que define un formato de transferencia de datos multiplataforma (Barrero, 2000). Sus principales propiedades son:

- Facilidad de implementación: tanto en la descripción de diferentes especificaciones de formatos como en su análisis.
- Por estar referido a nivel de la aplicación y no de programación, XML se considera un lenguaje de bajo nivel.
- XML ha nacido no solo para su aplicación en Internet, sino que se propone para el intercambio de información estructurada entre diferentes plataformas.
- XML hace uso de etiquetas y atributos y deja la interpretación de los datos a la aplicación que los utiliza.
- XML se considera un metalenguaje, *i.e.*, un lenguaje para especificar lenguajes.

Para verificar que los documentos estén bien formados se utiliza un DTD (definición de tipo de documento) (W3C XML Working Group, 2003). Se trata de una definición de los elementos que pueden incluirse en el documento XML, la relación entre ellos, sus atributos, posibles valores, etc.; en otras palabras, de la definición de las reglas gramaticales que especifican un tipo de documento dado, por ejemplo, las etiquetas y sintaxis de html. Como es de suponer, la última especificación de X3D presenta un DTD específico para el formato XML. Sin embargo, aún no se considera definitivo, puesto que presenta algunas deficiencias frente a la extensibilidad que X3D requiere.

[6] Al igual que para X3D, el primero de ellos (*phong*) cumple la función de indicar el nivel de brillo de la superficie, así como *specular* se encarga de indicar la capacidad de reflejar la luz que tiene la superficie del objeto.

Existen dos especificaciones del W3C para el procesamiento y análisis de documentos XML:

1. La DOM (Document Object Model) permite la creación de una estructura en árbol de los elementos del documento, es decir, organiza las etiquetas y atributos jerárquicamente, brindando a su vez funciones que permitan que cada nodo se pueda navegar, leer y modificar.
2. SAX (Simple API for XML) es aquella que permite a las aplicaciones manejar grandes documentos XML. En lugar de leer un documento por completo y mantenerlo en memoria (tal como hace DOM), SAX realiza una búsqueda sobre los datos leyendo y descartando elementos del documento, hasta que encuentra aquellos que cumplen un criterio específico.

La especificación más adecuada para el proyecto es DOM, pues permite cargar todo el documento, validarlo, recorrer todos sus elementos y de esta forma realizar una traducción sencilla entre los formatos. Se escogieron las librerías DOM Level 2 de QT de Trolltech, que se encuentran bajo licencia GPL de libre uso y permiten el acceso al código fuente (Trolltech, 2003; Lye, 2003).

Algoritmo de conversión

La entrada del algoritmo de conversión es un archivo con la escena especificada en X3D, su salida corresponde a un archivo con la escena descrita en POV-Ray. Luego de generar una cámara por defecto en el archivo de destino, se procede a buscar en preorden dentro del árbol X3D generado por DOM, los nodos de tipo *shape*, *pointlight*, *viewpoint* y *background* (de aquí en adelante llamados "nodos principales"), dado que estos representan las estructuras principales de POV-Ray. En el análisis de los nodos *shape* se tienen en cuenta dos posibles nodos hijos: *appearance,* para definir las características visuales de la figura (brillo, material, reflejo), y la propia figura que define sus dimensiones según su género (caja, cilindro, esfera, etc.). Es importante resaltar que las características visuales mencionadas residen como atributos del nodo "material", que en caso de existir se encuentra como hijo de *appearance*.

Cuando se han analizado los cuatro tipos de nodos principales mencionados anteriormente, se procede a buscar para cada uno de ellos (recorriendo el árbol en ascendencia jerárquica), si existen cero, uno o varios nodos padre de tipo *transform*. Lo anterior, debido a que el nodo *transform* define las modificaciones espaciales de sus hijos (escala, rotación y traslación), y si existen varios de estos nodos sobre un mismo nodo principal todas esas transformaciones se deben aplicar a este en el orden en que se encuentran al ascender en el árbol jerárquico.

En tal modelo, conforme con la obtención de los objetos principales declarados en el código XML bajo el estándar X3D con sus atributos y transformaciones, se genera simultáneamente el código para POV-Ray. El procedimiento en pseudocódigo se describe en el algoritmo 1.

Algoritmo 1: x3d2pov

Precondición: Archivo con la escena especificada en X3D
Postcondición: Archivo con la escena especificada en POV-Ray

```
(1)   escribir una structura camera con atributos por
      defecto en el archivo de POV-Ray
(2)   while existan nodos para analizar
(3)       if nodo es pointlight
(4)           leer atributos de la etiqueta XML
(5)           leer nodos transform que hereden sus
              atributos a pointlight
(6)           escribir la estructura light source en el
              archivo de POV-Ray efectuando la
              conversión de sintaxis y la agregación de
              transformaciones.
(7)       else if, nodo es viewpoint
(8)           leer atributos de la etiqueta XML
(9)           leer nodos transform que hereden sus
              atributos a pointlight.
(10)          escribir la estructura light source en el
              archivo de POV-Ray efectuando la
              conversión de sintaxis y la agregación de
              transformaciones.
(11)      else if nodo es background
(12)          leer atributos de la etiqueta XML
(13)          escribir la estructura background en el
              archivo de POV-Ray efectuando la
              conversión de sintaxis.
(14)      else if nodo es shape
(15)          leer nodo hijo de figura (box, cylinder,
              etc.)
(16)          if shape tiene hijo appearance
(17)          leer atributos del nodo material
                  debajo de appearance
(18)          buscar nodos transform que hereden sus
              atributos a shape
(19)          escribir la estructura de la figura
              correspondiente (box, cylinder, etc.) en el
              archivo de POV-Ray efectuando la
              conversión de sintaxis y la agregación de
              transformaciones y propiedades del
              material.
(20)  end while
```

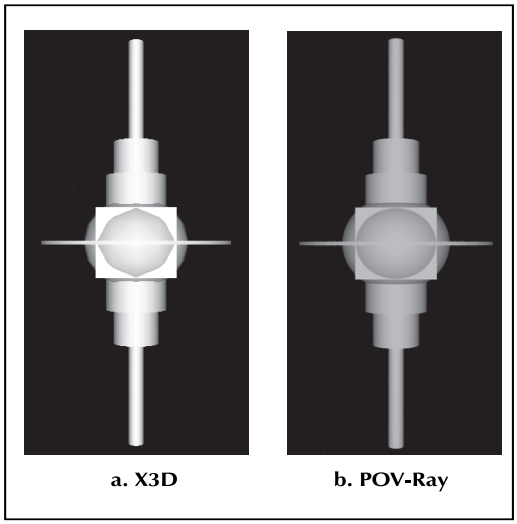
En la Tabla 5 se puede apreciar un escenario equivalente, descrito en ambos códigos, en donde se ha analizado el árbol X3D extrayendo las estructuras *background*, *pointlight* (*light source* en POV-Ray) y *shape* (para representar el objeto *cylinder*), con sus respectivos atributos.

Tabla 5. Escenario sencillo descrito en X3D y en código para POV-Ray

X3D	*POV Script*
<**Scene**> <**Background** groundColor="0 0 0"/> <**Transform** translation="0 20 0"> <**PointLight**/> </Transform> <**Shape**> <**Cylinder**/> </Shape> </Scene>	**background**{color<0.900,0.900,0.900> } **camera**{location<-9.939,-10.000,12.876> look_at <1.447,1.337,2.724> } **light_source** {<0, 20, 0> color rgb <1.000, 1.000, 1.000> } **cylinder** {<0,0,1>, <0,0,0>, 1 color rgb <1.000, 1.000, 1.000> }

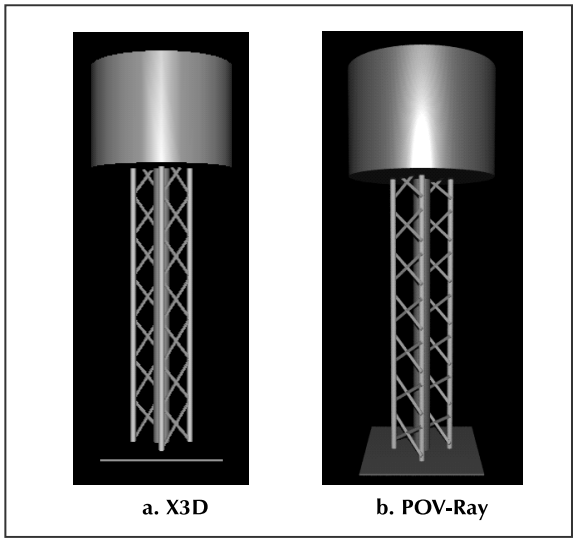
Resultados y trabajo futuro

En la actualidad, X3D2POV está en la capacidad de traducir y mapear primitivas básicas, transformaciones y demás elementos descritos en la especificación de objetos en X3D En las figuras 1 a 3 se presentan algunos resultados obtenidos. Se puede apreciar la calidad generada usando el trazador de rayos después de llevada a cabo la traducción de los objetos, frente a la obtenida mediante el *browser* VRML/X3D.



a. X3D b. POV-Ray

Figura 1. La imagen *a* es menos suavizada que *b*

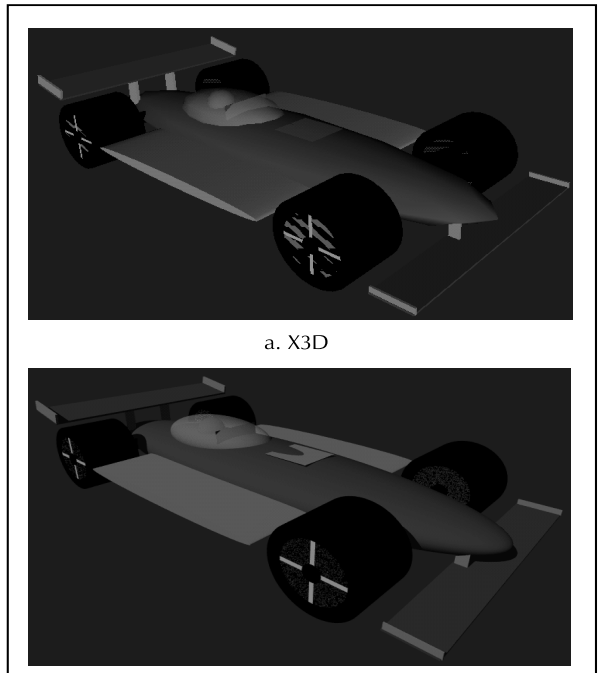


a. X3D b. POV-Ray

Figura 2. Nótese la calidad de renderización de las propiedades reflectivas de los objetos. Se nota un mejor manejo de reflejos en la imagen *b*

En conformidad con el avance en el proceso de especificación al que el estándar X3D está sometido, es importante guiar el desarrollo de la aplicación extendiendo su capacidad de interpretación, integrando soporte para lo que hace falta de la especificación actual de X3D; de igual manera, es vital seguir actualizando la aplicación para reflejar los cambios y ampliaciones que se vayan realizando a dicho estándar.

De otra parte, el problema inverso, la traducción de una escena descrita en POV-Ray a X3D, también resulta



a. X3D

b. POV-Ray

Figura 3. Nótese la diferencia en la calidad del dibujo de las transparencias e iluminación en el ambiente

atractivo y, de especial interés, cuando se busque efectuar su visualización interactiva mediante, por ejemplo, *browser* o *plug-ins*.

Bibliografía

Barbero, A., Tutorial XML., Revista Interamericana de Nuevas Tecnología de la Información. Bogotá., Vol. 3-4, Jul/Dic, 2000) pp. 14-23.

Brutzman, D., y Muñoz, G., Extensible 3D (X3D) Tooltips en Español Disponible Internet: http://www.web3d.org/TaskGroups/x3d/translation/X3dTooltipsSpanish.html Ultima Visita: Junio 30, 2003.

Costas, P., Ordax, S., y Selva, R., Documentación de POV-Ray en Castellano. Disponible en Internet: http://www.arrakis.es/~pcostas/povray/. Ultima Visita: Junio 30, 2003.

Holmes, S., The Availability of VRML Models on the Internet. Multimedia Systems Coursework, Department of Electronics and Computer Science, University of Southampton, Southampton, 2002.

Lye, G., Parsing XML with QT's DOM classes., Disponible en Internet: http://zez.org/article/articleview/28/ Ultima Visita: Junio 30, 2003.

POV TEAM. POV-Ray 3.5 Users Documentation. Persistence of Vision™ Ray Tracer (POV-Ray™) Disponible en Internet: http://www.povray.org/. Ultima Visita: Junio 30, 2003.

TROLLTECH. Qt Reference Documentation. XML Module. Disponible en Internet: http://doc.trolltech.com/3.1/xml.htm Ultima Visita: Junio 20, 2003.

W3C XML Working Group. XML specification DTD. Disponible en Internet: http://www.w3.org/TR/REC-xml. Ultima Visita: Junio 30, 2003.

Wald, I., Slusallek, P., State of the Art in Interactive Ray Tracing. Computer Graphics Group, Saarland University., The Eurographics Association. September 2001, pp. 21-42.

WEB3D Consortium. Extensible 3D (X3D) International Standard ISO/IEC 775:200x, Disponible en Internet: http:/ /www.web3d.org/TaskGroups/x3d/X3DSpec_CD_Preview/ index.html. Ultima Visita: Junio 30, 2003a.

WEB3D Consortium. Examples-Extensible 3D (X3D) Graphics. Disponible en Internet : http://www.web3d.org/ TaskGroups/x3d/translation/X3D-Examples.zip. Ultima Visita: Junio 30, 2003b.