

Un método de compresión de mallas basado en una representación simplicial

Andrés Adolfo Navarro Newball
anavarro@puj.edu.co

Francisco Julián Herrera Botero
herbo@puj.edu.co

Luis Eduardo Múnera Salazar
lemunera@icesi.edu.co

Fecha de recepción: 4-01-2008

Fecha de selección: 17-10-2008

Fecha de aceptación: 8-08-2008

ABSTRACT

Polygonal meshes, which approximate the surface of an object, are the most common method for representing three dimensional objects. Here, in order to attain efficiency, the triangle is the most used polygon. However, the number of required triangles increases as more realism is needed in a complex form. For this reason, mesh optimization techniques have been created that keep resolution and enhances performance. This way, the polygonal mesh occupies less memory space. This paper presents a mesh compression technique which is based on mathematical topology and the simplicial representation. This

technique was developed in order to compress meshes that were going to be sent through Internet.

KEYWORDS

Mesh, Simplicial Complex, Topology, Polygon, Compression.

RESUMEN

El método más común de representar objetos de tres dimensiones (3D) en computación gráfica es la malla poligonal, que aproxima la superficie del objeto utilizando polígonos. Aquí, por desempeño y eficiencia de procesamiento, el triángulo es el polígono más usual. Sin embargo, el número de triángulos requeridos para crear una

representación de una forma compleja aumenta en la medida en que se busca mayor exactitud o resolución. Por esta razón, se han creado técnicas de optimización de mallas que buscan disminuir el número de polígonos sin perder resolución. De esta manera, la malla poligonal puede ocupar menos espacio en memoria. Este artículo presenta una técnica de compresión de mallas que se basa en la teoría matemática de la topología y la repre-

sentación de símlices. Esta técnica fue creada con el fin de comprimir mallas que van a ser transmitidas a través de internet para ser usadas en aplicaciones interactivas a través de la red.

PALABRAS CLAVE

Malla, complejo simplicial, topología, polígono, compresión

Clasificación Colciencias: Tipo 1

I. INTRODUCCIÓN

Típicamente, los objetos de tres dimensiones se representan por superficies creadas a partir de polígonos que aproximan la representación de su forma. Entre las razones para representar un objeto mediante un modelo de superficie basado en polígonos están:¹

- Cuando el objeto es una superficie que se puede suponer sin grosor (por ejemplo, la chapa metálica del capó de un vehículo). Este tipo de representación permite visualizar superficies abiertas, mientras que los sólidos se caracterizarán por tener su superficie necesariamente cerrada sobre sí misma.
- Cuando sólo interesa visualizar el aspecto visual externo de un objeto, sin detalles sobre su estructura interna, aunque el objeto ocupe un cierto volumen.
- Cuando se realiza una visualización en tiempo real, y para ello se utiliza hardware o software gráfico que está preparado sólo para visualizar polígonos.

La generación de mallas juega papel fundamental en cualquier problema computacional científico en el cual la geometría de una región a modelar es compleja. Aun con las actuales herramientas de software, la generación de mallas se lleva una gran parte del esfuerzo necesario para resolver un problema. Así pues, la investigación acerca de la generación de mallas está principalmente dirigida hacia la elaboración de algoritmos que las generen de manera automática. Aquí el término automático se refiere a que el esfuerzo necesario por parte del usuario para generar una malla sea

reducido al mínimo. En algunos casos, la generación de mallas requiere de cálculos procedimentales complejos; por ejemplo, cuando se hace generación fractal de terrenos. En otros, requiere de habilidades de dibujo; por ejemplo, cuando se usa software de edición 3D como 3D Studio Max. La malla es la principal estructura de datos en la computación gráfica; así, se constituyen en una estructura fundamental dentro de mundos virtuales creados con técnicas de computación gráfica tales como el propuesto en los sistemas WESST-OT e IMCA.^{2,3,4}

A continuación, se enuncian algunas de las técnicas de optimización de mallas; luego, se describe qué es la topología y se habla de la representación simplicial de algunas mallas típicas. Seguidamente, se describe la forma de almacenar una matriz de símlices comprimida y se presentan pruebas sobre la técnica de compresión sugerida tomando el caso de la malla de un icosaedro. Finalmente, se exponen unas conclusiones.

2. OPTIMIZACIÓN DE MALLAS⁵

Existen técnicas de optimización de mallas que buscan disminuir el número de polígonos requeridos para representar una superficie sin afectar el nivel de resolución del objeto 3D desplegado. Estas, permiten crear una malla capaz de ocupar menos espacio en memoria y susceptible de ser desplegada o transmitida más rápidamente. A continuación se enuncian algunas:

2.1. Modelos de compresión poligonal.

Normalmente los procesadores gráficos no están limitados por los modelos poligonales, pero sí por el ancho de

banda al realizar la transferencia de información entre sub-sistemas. Esto significa que en general las tarjetas de tres dimensiones están en capacidad de renderizar más triángulos por segundo de los que pueden ser transportados por el bus de datos. Por ejemplo, para transferir una escena compleja que posee un millón de triángulos a 30 Hz y 36 bytes/triángulo se requiere un canal de 1080MB/seg entre el procesador y el hardware gráfico. Así, se proponen tres maneras de reducir la información por polígono: reducir la información enviada por el vértice del polígono; reducir el número de vértices, normalmente “tri-strips” al objeto poligonal o ; reducir el número de polígonos por objeto, conforme al número de píxeles que se proyecta.

2.2 Codificación de la conectividad.

La relación de conectividad en una malla normalmente se considera como distante. En algunas aplicaciones la conectividad es implícita o se sobreentiende; por ejemplo, al trabajar con mallas que representan terrenos, solo se necesita conocer el vértice y su altura; así, la información que no aporta a la conectividad es codificada. En contraste, el implementar los triángulos para definir la conectividad entre los vértices puede producir redundancia de información; aquí, esquemas como el de Rossingnac, que no hace uso explícito de la conectividad y no realizan duplicación de vértices permiten optimizar la malla. La reducción de costo se puede apreciar en los ítemes 1 y 2:

Esquema clásico:

$$6n \log_2(n) \text{ bits (1)}$$

Esquema Rossingnac:

$$6n + 5n \log_2(n) \text{ bits (2)}$$

El esquema de Rossingnac es de un costo mucho menor para objetos de más de 64 vértices.

2.3. Lista de triángulos

Las listas de triángulos o “tri-strips” son una manera de comprimir la información de conectividad de un vértice perteneciente a una malla de triángulos. Primero ordena los triángulos consecutivos que forman un borde, reduciendo los vértices en listas desde $3n$ (si n triángulos son enviados separadamente) a $n + 2$ ya que es necesario especificar sólo un nuevo vértice por triángulo. Sin embargo, este es un límite teórico inferior para un modelo complejo que solo se acerca a un algoritmo que encuentra buenos “tri-strips”.

2.4. Algoritmos locales y globales

Los algoritmos locales funcionan como un algoritmo de “tri-strips” para mallas. Aquí, se construye una ruta por medio de triángulos, para ello escoge un vecino que es actualmente el triángulo adyacente no recorrido con menor número de vecinos. Si se encuentra más de un triángulo con el mismo menor número de vecinos, se mira un nivel adelante y se aplica la misma verificación. Por otra parte, los algoritmos globales hacen un análisis global de una malla usando técnicas de “patchification” que dependen de la observación de muchos polígonos.

2.5. Vertex buffers

Trata de utilizar un buffer de memoria en el hardware gráfico para habilitar el re-uso de vértices comunes

a dos o más tri-strips. Los vértices que se reutilizan están contenidos en una pila.

2.6. Procesamiento a nivel de detalle

Los modelos de mallas poligonales están bien establecidos como el factor estándar de representación para gráficos por computadora. El nivel de detalle o número de polígonos necesarios se incrementa al sintetizar un objeto en una alta calidad o en un objeto complejo muy grande. Si el objeto es renderizado en la pantalla teniendo diferentes distancias de visualización, algunas de sus partes se verán como simples píxeles en la pantalla. Así, es adecuado almacenar versiones con mayor o menor resolución que van a ser desplegadas, lo cual depende de la distancia de visualización.

3. TOPOLOGÍA Y COMPLEJOS SIMPLICIALES BÁSICOS^{6,7}

La topología es una disciplina matemática que estudia las propiedades de los espacios topológicos y las funciones continuas. La topología se interesa por conceptos como proximidad, número de agujeros, el tipo de consistencia (o textura) que presenta un objeto, comparar objetos y clasificar, entre otros múltiples atributos donde destacan conectividad, compacidad y metricidad.

Una malla en tres dimensiones puede verse como un espacio topológico que admite una triangulación por un complejo simplicial. Las triangulaciones permiten analizar estos espacios combinatoriamente. Los complejos simpliciales quedan definidos por sus vértices y símlices; aquí, cada n-sím-

plice es un conjunto de $n+1$ vértices, y a cada complejo simplicial K se le asocia un espacio topológico o realización geométrica de K , que se construye pegando convexos determinados por los símlices. En concreto, un complejo simplicial K consiste en un conjunto de vértices V_K y un conjunto S_K cuyos elementos son subconjuntos finitos no vacíos de V_K (llamados símlices) con las siguientes propiedades:

- Todo vértice de K es un símlice (es decir, S_K contiene todos los subconjuntos de un elemento de V_K).
- Todo subconjunto no vacío de un símlice es un símlice (es decir, si $s \in S_K$ y $s' \in s$ es no vacío, entonces $s' \in S_K$ y se dice que s' es una cara de s).
- Si $s \in S_K$ tiene $n + 1$ elementos, decimos que s es un n -símlice ó, equivalentemente, que $\dim s = n$. Por lo tanto los vértices son los 0-símlices de K .

Los complejos simpliciales se suelen representar por medio de una matriz de incidencia (MI), muy conveniente para cálculos computacionales, cuyas columnas son etiquetadas por sus vértices, y las filas son etiquetadas por los símlices. A continuación se muestran los complejos simpliciales con sus respectivas MI. La Figura 1 revela un punto representado como un símlice de dimensión cero. La Figura 2 enseña una línea (arista) representada como un símlice de dimensión 1. La Figura 3 muestra un triángulo como un símlice de dimensión dos. Para la mayoría de los casos se puede tomar al triángulo como el elemento constitutivo de las mallas que aproximan las superficies

en computación gráfica. La Figura 4 muestra un tetraedro sólido como un símplice de dimensión tres. Para la mayoría de los casos, se puede tomar al tetraedro como el elemento consti-

tutivo de los volúmenes. Este artículo sólo se enfoca en la representación 3D de superficies, por lo que el complejo simplicial tetraedro se brinda más a manera de información.

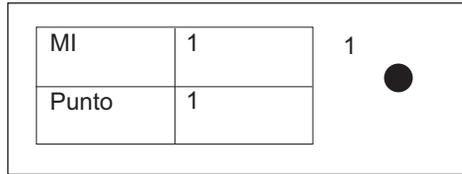


Figura 1.
Complejo simplicial punto

Figura 2.
Complejo simplicial línea

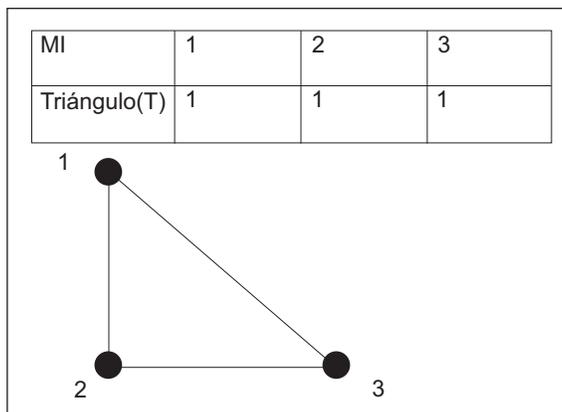
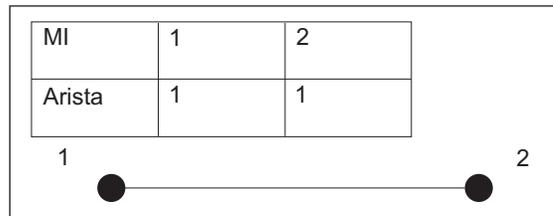
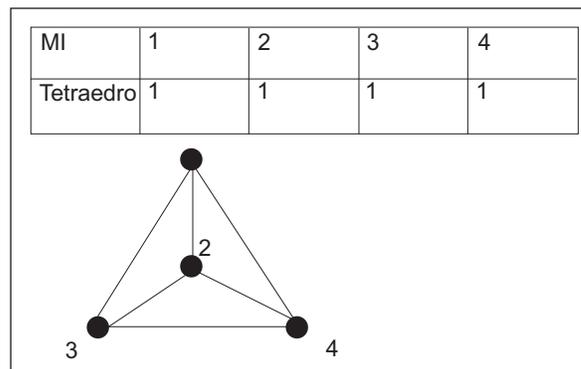


Figura 3.
Complejo simplicial triángulo

Figura 4.
Complejo simplicial tetraedro sólido.



La Figura 5 y la Tabla 1 muestran la malla de la superficie de un cubo representado a partir de triángulos representados como símplices de

dimensión 2. Esta sería la representación mediante su matriz de incidencia.

Tabla 1. Complejo simplicial para la malla de un cubo

MI	1	2	3	4	5	6	7	8
T1	1	1	1					
T2		1	1	1				
T3		1		1				1
T4		1				1		1
T5					1	1		1
T6					1		1	1
T7	1				1		1	
T8	1		1				1	
T9	1	1			1			
T10		1			1	1		
T11			1	1			1	
T12				1			1	1

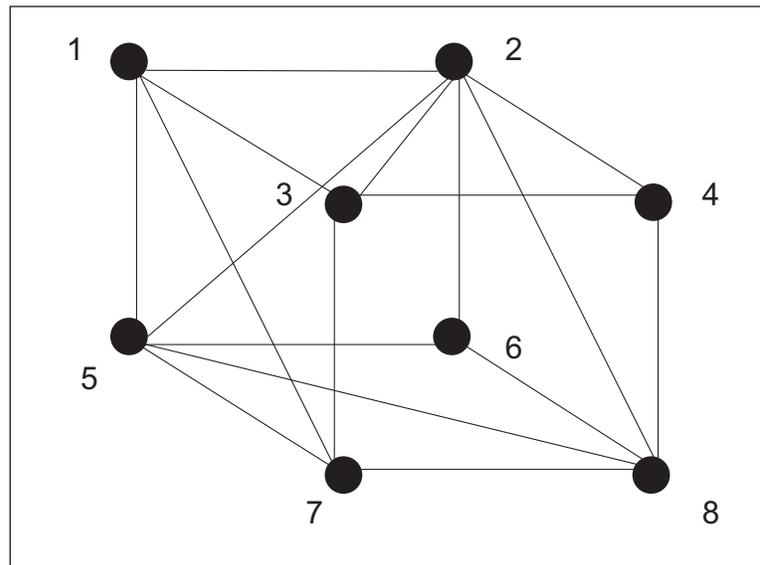


Figura 5. Complejo simplicial de la malla cubo

4. TÉCNICA DE COMPRESIÓN PROPUESTA

La idea de esta técnica de compresión nace de la necesidad de transmitir mallas en tres dimensiones del entorno de simulación WESST- OT a través de internet. WESST – OT es un sistema que ofrece la posibilidad de practicar habilidades básicas de cirugía endoscópica en otorrinolaringología, a través de la utilización de técnicas de computación gráfica y rea-

lidad virtual. Se ha evidenciado que el tamaño de las mallas desplegadas afecta directamente el desempeño de la aplicación. A continuación se describe el desarrollo de la técnica de compresión valiéndose del ejemplo de la malla de un icosaedro.

El archivo icosaedroAbril15-2005.wrl de 803 bytes contiene una definición de la malla de un icosaedro en VRML (Figura 6). Para mayor claridad referirse a la Nota 8

```
#VRML V2.utf8
Shape {
  geometry IndexedLineSet {
    #Coordenadas de los vértices
    coord Coordinate {
      point [
        0 0 1.175571, 1.051462 0 0.5257311,
        0.3249197 1. 0.5257311, -0.8506508 0.618034 0.5257311,
        -0.8506508 -0.618034 0.5257311, 0.3249197 -1. 0.5257311,
        0.8506508 0.618034 -0.5257311, 0.8506508 -0.618034 -0.5257311,
        -0.3249197 1. -0.5257311, -1.051462 0 -0.5257311,
        -0.3249197 -1. -0.5257311, 0 0 -1.175571,
      ]
    }
    #Polígonos
    coordIndex [
      0,1,2, -1, 0,2,3, -1,0,3,4,-1,0,4,5,-1,0,5,1,-1,1,5,7,-1,1,7,6,1,
      1,6,2,-1, 2,6,8,-1,2,8,3,-1,3,8,9,-1,3,9,4,-1,4,9,10,-1,4,10,5,1,
      5,10,7,-1,6,7,11,-1,6,11,8,-1, 7,10,11,-1,8,11,9,-1,9,11,10,-1,
    ]
  }
}
```

Figura 6. Definición de la malla en VRML

De esta definición de malla se tomaron las coordenadas de los vértices en un archivo aparte, pues inicialmente la representación simplicial no soporta valores distintos a cero o uno;

además, la definición de polígonos se convirtió a una representación como la mostrada en la Tabla 2 que fue almacenada en el archivo icosaedro.in de 520 bytes.

Tabla 2. Complejo simplicial icosaedro, archivo icosaedro.in.

MI	1	2	3	4	5	6	7	8	9	10	11	12
T1	1	1	1	0	0	0	0	0	0	0	0	0
T2	1	0	1	1	0	0	0	0	0	0	0	0
T3	1	0	0	1	1	0	0	0	0	0	0	0
T4	1	0	0	0	1	1	0	0	0	0	0	0
T5	1	1	0	0	0	1	0	0	0	0	0	0
T6	0	1	0	0	0	1	0	1	0	0	0	0
T7	0	1	0	0	0	0	1	1	0	0	0	0
T8	0	1	1	0	0	0	1	0	0	0	0	0
T9	0	0	1	0	0	0	1	0	1	0	0	0
T10	0	0	1	1	0	0	0	0	1	0	0	0
T11	0	0	0	1	0	0	0	0	1	1	0	0
T12	0	0	0	1	1	0	0	0	0	1	0	0
T13	0	0	0	0	1	0	0	0	0	1	1	0
T14	0	0	0	0	1	1	0	0	0	0	1	0
T15	0	0	0	0	0	1	0	1	0	0	1	0
T16	0	0	0	0	0	0	1	1	0	0	0	1
T17	0	0	0	0	0	0	1	0	1	0	0	1
T18	0	0	0	0	0	0	0	1	0	0	1	1
T19	0	0	0	0	0	0	0	0	1	1	0	1
T20	0	0	0	0	0	0	0	0	0	1	1	1

Luego, se estudió y evolucionó una técnica de compresión de la nueva representación. Este proceso se explica en las siguientes secciones.

4.1. Propuesta I

Contiene la primera propuesta para comprimir la matriz de simples. Para esto se debe:

- Convertir la matriz de incidencia a decimal.
- Cada fila de la matriz (cara de la malla) indica con “1” qué vértice está utilizando.
- Se debe almacenar un decimal que indicará la posición o columna de la matriz en la que se encuentra el vértice a utilizar.

Así, se logra comprimir el archivo de entrada icosaedro.in a 175 bytes

4.2. Propuesta II

Consiste en representar cada fila de la matriz con un solo número decimal, ya que la matriz es de ceros y unos.

Así, se logra comprimir el archivo de entrada icosaedro.in a 121 bytes.

4.3. Coordenadas de los vértices

Adicionalmente, se decidió separar la información espacial de los vértices en el archivo CoordenadasPuntos.txt de 303 bytes.

4.4. Primera comparación

La Tabla 3 muestra la comparación de los primeros dos métodos de compresión propuestos.

Tabla 3. Primera comparación de métodos

	Archivo	Propuesta I	Propuesta II
Sin compresores externos	Archivo VRML original. Total: 803 bytes	175 bytes – representación simplicial comprimida 303 bytes – Archivo de coordenadas Total: 478 bytes	121 bytes – representación simplicial comprimida 303 bytes – Archivo de coordenadas Total: 424 bytes
Usando compresión .ZIP	Total: 442 bytes	175 bytes – representación simplicial comprimida (sin .Zip) 232 bytes – Archivo de coordenadas comprimido con .Zip Total: 407 bytes	175 bytes – representación simplicial comprimida (sin .Zip) 232 bytes – Archivo de coordenadas comprimido con .Zip Total: 353 bytes
Observaciones		La representación simplicial comprimida crece a 183 bytes cuando se le aplica .Zip, por eso esto se descarta	La representación simplicial comprimida crece a 179 bytes cuando se le aplica .Zip, por eso esto se descarta

Es claro, de acuerdo con los datos de la Tabla 3 que la Propuesta II de compresión es mejor en términos del tamaño del archivo de salida

resultante. La Tabla 4 muestra las ventajas y desventajas de las propuestas I y II.

Tabla 4. Ventajas y desventajas de las propuestas de compresión

	Propuesta I	Propuesta II
Desventajas	Si la cadena de 0s y 1s en la matriz de simples es muy grande, se gasta mas al indicar la ubicación de un 1 dentro de la matriz	Aparentemente poco escalable pues el número de vértices de una malla puede variar. En caso de que existan muchos vértices cada fila de la matriz no se podría representar como un decimal
Ventajas	Muy sencillo de almacenar La descompresión es inmediata	La compresión es sencilla En las pruebas realizadas en un archivo pequeño la compresión fue la mejor

De acuerdo con los resultados mostrados se optó por encontrar una forma de mejorar la Propuesta II, ya que esta fue la que evidenció mayor potencial de compresión. La siguiente sección explica las mejoras realizadas sobre la técnica Propuesta II.

5. MEJORAS A LA PROPUESTA II DE COMPRESIÓN

Se sabe que un número entero utiliza 32 bits, esta propuesta trata de aprovechar al máximo ese espacio con el mínimo desperdicio. A continuación se describe una técnica de

tres etapas para la compresión de la representación simplicial de una malla que mejora la propuesta en la sección 4.2. Se toma nuevamente el ejemplo del icosaedro como apoyo a la explicación.

5.1. Primera etapa

Para comenzar se toma el complejo simplicial icosaedro de la Tabla 2 y se

obtiene una representación de cadenas de 32 bits. La Tabla 5 muestra la nueva representación del icosaedro. Esta representación debe almacenar el número de vértices de la malla, pues es la única manera de recordar cuántas columnas tiene la matriz de simples original.

Tabla 5. Representación del complejo simplicial en cadenas de 32 bits

Cadena 1	11100000000010110000000010011000
Cadena 2	00001000110000001100010000000100
Cadena 3	01010000010000110000011000100000
Cadena 4	00100010100000110000100000010000
Cadena 5	11000001100001000000100001100000
Cadena 6	11000010000001010010000000110001
Cadena 7	00000010100100000001001100000000
Cadena 8	11010000000001110000000000000000

5.2. Segunda etapa

En esta etapa se definen:

@: Carácter de inicio de la malla.

V: Número de vértices de la malla. Es decir, número de columnas de la matriz de simples.

#: Carácter de inicio de número.

N_i : i-ésimo número hexagesimal.

Una vez ejecutada esta etapa el icosaedro quedaría representado por 8 números hexagesimales, 8 caracteres de inicio de número (#) y, un carácter de inicio de malla (@) seguido de un decimal que indica el número de vértices, tal como se muestra en el ítem 3

@12#E00B0098#8C0C404#50430620
#22830810#C1840860#C2052031#29
01300#D0070000 (3)

De esta manera, el espacio ocupado por la malla sería el mostrado en el

ítem (4). Es decir, que se logra una reducción de más del 40% respecto a la propuesta II inicial.

$(16 \text{ bits} \times (17) + 32 \text{ bits} \times (9)) / 8 \text{ bytes}$
/ bits = 70 bytes. (4)

5.3. Tercera etapa

Aunque no es el caso del icosaedro, en una malla suficientemente grande puede ocurrir que dos o más hexagesimales producidos durante la segunda etapa del proceso de compresión sean cero. Así, la cadena resultante estaría brindando información redundante. Por eso, cuando aplique, se debe indicar el número de ceros consecutivos a la cadena producida en la etapa 2 de una forma abreviada. A continuación (ítem 5) se presenta una malla en proceso de compresión que incluye varios ceros consecutivos y que ocupa 70 bytes.

@12#0#0#0#0#8C0C404#50430620
#22830810#C1840860 (5)

Esta podría ser representada mejor como se indica en el ítem 6:

@12\$4#8C0C404#50430620
#22830810#C1840860 (6)

Aquí:

\$: indica el origen de una secuencia de ceros, seguido del número de ceros

Y, el tamaño ocupado por la malla sería el mostrado en el ítem 7:

$(32 \text{ bits} \times (6) + 16 \text{ bits} \times (6)) / 8 \text{ bytes/ bits} = 36 \text{ bytes. (7)}$

Esto significa una reducción del 70% respecto a la propuesta original y del 49% respecto a haber terminado la compresión en la etapa 2.

6. CONCLUSIONES Y TRABAJO FUTURO

El algoritmo de compresión en formato .zip, comprime a mayor tamaño cuando el archivo es muy pequeño, pero comprime mejor cuando el archivo es grande. Para hacer una comparación justa entre los métodos y .zip es necesario utilizar una malla más grande que un icosaedro.

Las matrices de símlices no aportan en la transmisión de las coordenadas de los vértices; éstas se transmiten de igual forma en cualquier formato. En los métodos propuestos solo se quitan unas comas (,) originales del formato VRML. Un método de compresión como .zip puede aportar en la transmisión de vértices. Es necesario realizar estudios más profundos.

La representación de matrices de símlices de una malla se eligió como una alternativa a las estructuras de datos tradicionales por su simplicidad para representar la topología de los objetos 3D; además, tiene el potencial de explorar la utilización

de conceptos relacionados con base de datos (Nota 9). La utilización de estas teorías podría dar como resultado una técnica eficiente de descompresión y/o despliegue. A la fecha, el trabajo se ha enfocado principalmente en el mecanismo de compresión. Se debe crear una técnica de descompresión que sea rápida, sin embargo, dadas las características del método, esto puede ser sencillo.

El módulo de compresión de mallas propuesto busca hacer que el sistema, por sus mismas características gráficas, sea eficiente a nivel remoto, esto hace que se determine un método adecuado para implementación. Además, al manejar dentro del sistema de compresión de mallas un modo de optimización, se espera que el nivel de realismo no se pierda a medida que los objetos sean cada vez más complejos y "reales". La técnica propuesta aquí debe evolucionar para brindar información de aspectos como iluminación y materiales.

En un futuro se espera estudiar las técnicas tradicionales de optimización y representación de mallas y compararlas con la técnica propuesta, además de ver su posible aplicación a la misma.

7. BIBLIOGRAFÍA

1. J.D. Foley, A. Van Dam, S.K. Feiner, J.F. Hughes, Computer Graphics: Principles and Practice, Addison .Wesley, 1996.
2. A.A. Navarro, C.J. Hernández, J.A. Vélez B, L.E. Múnera, G.B. García, C.A. Gamboa, A.J. Reyes, Virtual surgical tele-simulations in otolaryngology, Studies in Health Technology and Informatics, vol. 111. pp. 353-355, Jan. 2005.

3. A.A. Navarro, F.J. Herrera, C.A. Marín, Using an interactive module to enhance and understand 3D cavity navigation: a patient's view, *Journal of Telemedicine and Telecare*, vol. 13, sup. 1, pp. 13-15, Jul. 2007.
4. A.A. Navarro Newball, F. Roviello, D. Prattichizzo, F.J. Herrera, C. Marín, Development of an interactive module to enhance and understand 3D cavity navigation, *Studies in Health Technology and Informatics*, vol 125, pp. 340-343, Jan. 2007.
5. A. Watt, F. Policarpo, 3D Games: Real Time Rendering and Software Technology, vol. 1, Addison Wesley, 2001.
6. J.R. Munkres, *Elements of Algebraic Topology*, Addison Wesley, 1984.
7. M. Henle, *A Combinatorial Introduction To Topology*, Dover Publications, 1994.
8. A.L. Ames, D.R. Nadeau, J.L. Moreland, *VRML 2.0 Sourcebook*, Wiley, 1996.
9. R. Dechter, Decomposing a Relation into a tree of Binary Relations, *Journal of computer and system sciences*, vol. 41, pp. 2-24, 1990.

CURRÍCULOS

Andrés A. Navarro Newball. Ingeniero de Sistemas y Computación egresado de la Pontificia Univer-

sidad Javeriana en Cali, tiene un MSc in Computer Graphics and Virtual Environments de la Universidad de Hull en Inglaterra y una especialización en Redes y Comunicación de la Universidad Icesi en Cali. Además, formó parte del Centro de Telemedicina de Colombia y fue profesor hora cátedra de la Universidad Icesi. Actualmente se desempeña como profesor de la Pontificia Universidad Javeriana – Cali, donde coordina el grupo de investigación *Destino* y está realizando estudios de PhD en la Universidad de Otago, Nueva Zelanda.

Francisco J. Herrera Botero.

Ingeniero de Sistemas y Computación, egresado de la Pontificia Universidad Javeriana en Cali, tiene una especialización en Redes y Comunicación (CCNA CISCO) de la Pontificia Universidad Javeriana - Cali. Se desempeñó como Asistente de Investigación (Programa Jóvenes Investigadores de Colciencias) en la Pontificia Universidad Javeriana – Cali. Además es fundador y gerente de la empresa de base tecnológica Soluciones Virtuales. Actualmente hace parte del grupo de investigación *Destino* y es profesor hora cátedra de la Universidad Icesi.

Luis. E. Múnera. Ph.D en Inteligencia Artificial de la Universidad Politécnica de Madrid y docente de la Universidad Icesi en Cali. ☀