TRABAJO ESPECIAL DE GRADO

“IMPLEMENTACIÓN DE UN SISTEMA DE MODELADO DE SÓLIDOS MEDIANTE OPERACIONES BOOLEANAS, CON APLICACIONES EN LA INGENIERÍA MECÁNICA”

Presentado ante la ilustre
Universidad Central de Venezuela
Por el bachiller: Ernesto F. Centeno P.
Para optar al Título de
Ingeniero Mecánico

Caracas, 2009

TRABAJO ESPECIAL DE GRADO

"IMPLEMENTACIÓN DE UN SISTEMA DE MODELADO DE SÓLIDOS MEDIANTE OPERACIONES BOOLEANAS, CON APLICACIONES EN LA INGENIERÍA MECÁNICA"

TUTOR ACADÉMICO: Prof. Antonio Barragán.

Presentado ante la ilustre
Universidad Central de Venezuela
Por el bachiller: Ernesto F. Centeno P.
Para optar al Título de
Ingeniero Mecánico.

Caracas, 2009

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE INGENIERIA
ESCUELA DE INGENIERIA MECANICA
DEPARTAMENTO DE DISEÑO

Caracas, 03 de julio de 2.009

ACTA

Los abajo firmantes, Miembros del Jurado designado por el Consejo de Escuela de Ingeniería Mecánica, para evaluar el Trabajo Especial de Grado presentado por el bachiller:

ERNESTO CENTENO

Titulado:

"IMPLEMENTACION DE UN SISTEMA DE MODELADO DE SÓLIDOS MEDIANTE OPERACIONES BOOLEANAS, CON APLICACIONES EN INGENIERIA MECANICA"

Consideran que el mismo cumple con los requisitos exigidos por el Plan de Estudios conducente al Titulo de Ingeniero Mecánico.

Prof. Rodolfo Berrios
Jurado

Prof. Manuel Martínez
Jurado

Prof. Antonio Barragán
Tutor

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE INGENIERIA
ESCUELA DE INGENIERIA MECANICA
DEPARTAMENTO DE DISEÑO

Caracas, 03 de julio de 2.009

Ciudadano
Prof. GERARDO RAMÍREZ
Jefe de División de Control de Estudios

Presente.-

Quienes suscriben Miembros del Jurado Examinador, designado por el Consejo de la Escuela de Ingeniería Mecánica, nos dirigimos a usted, con la finalidad de informarle se hemos decidido otorgarle al Bachiller:

ERNESTO CENTENO C.I.V.- 13.833.676

"MENCIÓN HONORÍFICA"

Por la excelencia demostrada en la realización del Trabajo Especial de Grado titulado:

"IMPLEMENTACION DE UN SISTEMA DE MODELADO DE SÓLIDOS MEDIANTE OPERACIONES BOOLEANAS, CON APLICACIONES EN INGENIERIA MECANICA"

Sin otro particular a que hacer referencia, quedamos de usted,

Prof. Rodolfo Berrios
Jurado

Prof. Manuel Martínez
Jurado

Prof. Antonio Barragán
Tutor

"Hacia el 50ª Aniversario del 21 de noviembre de 1957, Día del Estudiante"

INDICE

INDICE DE FIGURAS

INTRODUCCIÓN

El avance tecnológico vivido en los últimos años, ha llevado a dar grandes pasos en lo que respecta a la preparación de todos los profesionales en sus casas de estudio, en donde se deben y es obligación tener al día todos los conocimientos impartidos, especialmente en lo que respecta al ámbito de las nuevas tendencias de las tecnologías, actualmente la UCV, con la ayuda de todos los profesionales que laboran como docentes e investigadores, permiten abrir nuevas puertas para dichas tecnologías, colocando al estudiantado en el camino de dichas corrientes para su explotación y posterior desarrollo.

Uno de los temas que se podría decir que tiene mayor campo, es el desarrollo de modelos computacionales que permitan con la ayuda de programas de alta complejidad, simular o desarrollar modelos complejos para predecir su comportamiento; la base de todos estos conocimientos radica en respondernos un par de preguntas bien sencillas:

1) ¿Como se verá el objeto en la computadora?
2) ¿Puedo manipularlo a mi antojo una vez que lo tengo programado?
3) ¿Puedo simular el comportamiento de ese objeto bajo acciones externas como por ejemplo cargas?

El hecho de poder ver el objeto en el computador, es la consecuencia de un desarrollo complejo con una serie de operaciones y estándares que permiten al computador procesar la información y mediante códigos de programación, mostrar las intenciones del usuario respecto al sólido que desea trabajar, con esto hay que

recalcar que es importante diferenciar con el segundo punto, si el objetivo es únicamente graficar el sólido, se puede decir que aquí termina el proceso, pero si se plantea entrar en la respuesta de la segunda inquietud, se debe entonces indicar que no es solo un programa de graficación de sólidos sino un Programa de Modelado de Sólidos.

La complejidad de la programación y la funcionabilidad van de la mano con todos aquellos que colaboran en el desarrollo del programa, y esto en conjunto con satisfacer las necesidades del usuario, permite a través del sistemas UNIX que cualquier usuario tenga acceso al código fuente de dicho programa y permita su modificación en los detalles que así disponga, sin la necesidad de recurrir a la casa matriz que desarrolló el programa (software).

Desde hace cierto tiempo la Escuela de Ingeniería Mecánica de la UCV se encuentra encaminada a ubicar entre sus recursos, una plataforma que permita el Modelado de Sólidos (SMS) con la gran ventaja del desarrollo por y para estudiantes de la Escuela; los trabajos que ha venido desarrollando un grupo de alumnos de la Escuela de Ingeniería Mecánica a cargo del Prof. Antonio Barragán, han permitido el avance exitoso de dicho programa, el cual es poco a poco completado por todos aquellos que somos testigos de dicho proyecto.

Uno de los eslabones más importantes que se puede mencionar es el de las Operaciones Booleanas, aspecto que permite de muchas formas las operaciones con los sólidos que se quiere trabajar a partir de sólidos base, previamente programados.

El desarrollo de las Operaciones Booleanas se realiza bajo los mismos estándares que se ha venido trabajando el modelado gráfico y operaciones de creación de sólidos a través de métodos que se mencionarán más adelante, la parte de la interfaz gráfica, ya creada, permite al programador modificarla para así añadir las

funciones que se están agregando al programa macro, esto con el fin de la complementación de dicho software.

La programación en vista que será desarrollada para su uso bajo el sistema UNIX (Linux), debe ser realizada en Lenguaje C, por lo que se sigue el estándar de programación AINSI / ISO C99, con una metodología de trabajo que bajo la guía del Profesor Antonio Barragán, permite un esquema uniforme en todos los módulos que se anexen a dicho programa.

El impacto que se pretende tendrá dicho software es de gran importancia no sólo a nivel educacional sino que pudiera ser adaptado a procesos en la industria, ya que las personas que estamos viviendo dicho desarrollo estamos aquí, formándonos en esta área y pudiéramos implementar esta herramienta en un futuro cercano en nuestro desempeño profesional.

CAPITULO I

ANTECEDENTES

Para los años 50, en el primer sistema CAD al que se hace referencia [I1], se menciona el sistema gráfico SAGE (Semi Automatic Ground Enviroment) un sistema de defensa que permitía visualizar los datos de radar para propósitos defensivos ya a comienzos de los 60, según la misma referencia; se comenzaron a implementar los primeros sistemas gráficos interactivos CATIA y CADLink así como el desarrollo de los primeros estudios de la geometría proyectiva, y no es sino a partir de los años 70 que se desarrollan sistemas en 2-D para el dibujo HVAC (Heating, Ventilation and Air Conditioning, a final de los 70 [I1] se llegó al desarrollo de curvas mediante técnicas de aproximación denominadas B-spline (ver definiciones) o también denominadas NURBS y las superficies de Bézier. En este mismo periodo se implantan las bases teóricas del modelado de sólidos, con el desarrollo de los teoremas, métodos y técnicas fundamentales; todo esto con el fin de dar un comienzo al inicio de modelado de sólidos con representación en un sistema de 3-D. La empresa Computervision lanza la primera terminal de CAD con Gráficos Raster (ver definiciones) [I2].

Múltiples inconvenientes se presentaron a la hora de su desarrollo gráfico, uno de ellos la ambigüedad (que puede representar varias formas a la vez) o incluso el presentar objetos sin sentido (modelos que no pueden existir físicamente) de nuevo era requerida mas información por parte del usuario para su posterior desarrollo y que el sistema pudiera procesar la información de manera lógica.

En la posterior década, a mediados de los 80, se corrigen las deficiencias de los sistemas 3-D, aproximándose a los de modelado de sólidos. Los modelos de sistemas de superficies curvadas son ampliados y se proporciona una representación más estética y funcional, desarrollados por empresas como Autodesk; promoviendo la investigación y la mejora en los GIS (Geographical Information Systems). Es entonces en la década de los 80 [I2] que surge la sustitución masiva de las mesas de dibujo por sistemas computarizados CAD´s para su implementación en la industria.

El desarrollo de dichos programas modeladores de sólidos en lo que la Escuela de Ingeniería de la UCV respecta, han sido creados para resolver problemas con aplicación en la Ingeniería Mecánica, entre los cuales; se puede citar:

-Jaén y López (1990): Desarrollaron un sistema de post procesamiento gráfico para representar resultados de esfuerzos en órganos dentarios bidimensionales mediante el método de elementos finitos y resolver problemas de elasticidad. Se implementaron técnicas de representación de superficies isoparamétricas mediante funciones de interpolación.

-Alonzo y Oramas (1991): las investigaciones llevadas a cabo por estos autores llevaron al desarrollo de la representación gráfica de entidades bi y tridimensionales, aplicables a componentes mecánicos y de bioingeniería, y que permite realizar operaciones de rotación, traslación proyecciones, cambio de escala y visibilidad.

-Ordaz y Pulido (1992): centraron su investigación en el desarrollo de problemas geométricos asociados a la generación automática de mallas tridimensionales de elementos finitos en piezas de forma libre; centraron su estudio en enfocar la posición de un punto respecto a un sólido, un sólidos (elemento finito) respecto a un plano, intersecciones entre plano y plano, plano y sólido.

-Uzcátegui y Uzcátegui (1993): propusieron un programa con aplicación en la Ingeniería Mecánica en donde se implementan un conjunto de técnicas básicas de modelado de sólidos, obteniendo la representación de sólidos sencillos por medio del uso de operaciones constructivas (operadores de Euler) que permitieron crear herramientas (como la división de un sólido por un plano) y esto fue el inicio de la creación de sólidos complejos.

-Palma y Santaella (2005): El trabajo desarrollado, propone una base de datos para geometrías dentarias en modelos bidimensionales y tridimensionales que se puede ampliar mediante información proveniente de rayos X o tomografías axiales, para su posterior análisis mediante la implementación de métodos de elementos finitos. Las imágenes bidimensionales serían alimentadas por data proveniente de los resultados de los rayos X mientras que la data proveniente de la tomografía axial cumple con los parámetros para el desarrollo tridimensional.

-Martín H. Miguel A.(2007): desarrolló un programa de post procesamiento de datos resultantes del análisis por métodos numéricos de problemas de ingeniería, implementando un módulo de visualización gráfica de resultados, mediante el cual el usuario puede mostrar una representación graficadle estudio ya sea para analiza, interpretar, corregir y/o publicar los resultados obtenidos

-Chevron M. Laurent (2008): establece mediante la implementación de algoritmos de mediano nivel, el desarrollo y la generación de modelos sencillos como lo son: esfera, conos, paralelogramos, entre otros; se desarrolla también la aplicación de barrido rotacional y traslación que permite entonces a partir de una superficie, crear un arco o una columna.

CAPITULO II

MODELADO DE SÓLIDOS

Para establecer las bases del modelado de sólidos, es importante diferenciar el modelado de sólidos respecto a la representación gráfica, ya que esta última se enfoca en crear el dibujo de cualquier objeto, mientras que el modelado de sólidos permite a través del modelado geométrico la representación “completa” de un objeto físico sólido cualquiera; luego los modelos de superficies ofrecen información detallada de una superficie pero pueden no siempre proporcionar la información suficiente que permitan determinar las propiedades geométricas del objeto envuelto por la superficie

Un sistema de modelado de sólidos debe ser capaz de definir la apariencia de un objeto, la intersección del mismo con otros y en el caso de tener aplicaciones (en nuestro caso aplicaciones en la ingeniería mecánica) los procesos de manufactura podrían ser relacionados con dicho sólido.

Sistema de Modelado de Sólidos

Mäntylä [MANT88] expone entre sus trabajos que la idea fundamental es separar el modelo geométrico de la aplicación y desarrollar técnicas de modelado que sean relativamente independientes del objeto. Con esto, el sistema de modelado tiene como fin crear sólidos modelo de objetos físicos, y dar respuesta a preguntas geométricas.

Los problemas de Completitud, integridad, complejidad son resueltos en dicho modelador y representan una amenaza para el funcionamiento, así expone Chevron [CHEV08] en su desarrollo.

Entre las investigaciones de Mäntylä [MANT88] se exponen los componentes funcionales de un modelador de sólidos y el papel que cumple en el sistema geométrico, se muestra a continuación la figura

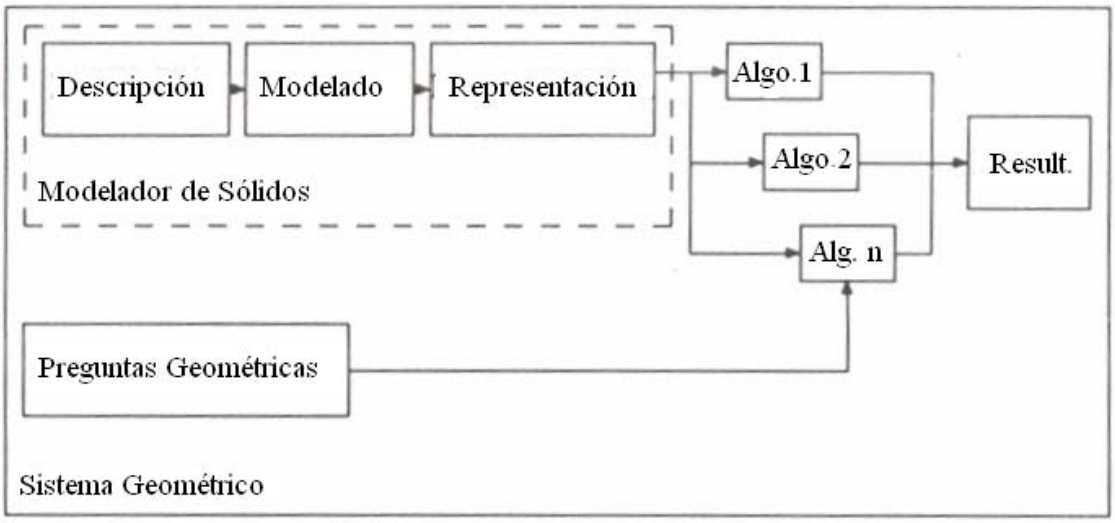


Figura 1. Componentes de un Sistema de Modelado de Sólidos. [MANT88]

Inicialmente, los objetos son descritos por el modelador en términos de un lenguaje descriptivo basado en los conceptos de modelado disponibles en el modelador de sólidos. El usuario debe introducir la descripción sencillamente como texto simple, o preferiblemente a través de una interfaz de usuario con la ayuda de interacción gráfica.

Una vez los datos ingresados, las descripciones del objeto son traducidas para crear la representación actual interna almacenada por el modelador. La relación entre el lenguaje descriptivo y la representación interna no necesariamente debe de ser directa:

las representaciones internas pueden emplear conceptos del modelado distintos a la descripción original. Más aún, un modelador de sólidos puede perfectamente incluir varios lenguajes descriptivos que se adapten a diferentes tipos de usuarios y aplicaciones.

El modelador debe proveer interfaces para comunicarse con otros sistemas. Esas interfaces son utilizadas para transmitir información dirigidas a varios algoritmos, o suministrar la información completa de modelos sólidos para otros sistemas de diseño.

El modelador debe también incluir facilidades para almacenar la descripción de objetos y demás datos, almacenados en permanentes bases de datos.

Con el propósito de dar respuesta a preguntas geométricas, una transformación desde la descripción externa hacia la representación interna es necesaria. De hecho, para aumentar la eficiencia de un modelador de sólidos, este debería soportar múltiples representaciones internas de los objetos; se tienen que incluir algoritmos de conversión que permitan modificar data desde una representación hacia otra.

Como introducción a los sistemas de modelado de sólidos debemos tener presente los siguientes conceptos para a la hora de elegir el sistema de desarrollo de las figuras, no entremos en errores de concepto y evitar en lo posible pasos innecesarios en el desarrollo deseado, los conceptos listados a continuación presentan un esbozo de las características principales de cada uno de los sistemas.

Esquemas de representación:

Representación de Frontera (B-Rep):

Es la abreviatura del término *Boundary Representation*, describe a un objeto en función de sus fronteras superficiales, es decir de las caras y superficies que lo configuran, esta definido por los vértices y aristas (pueden ser planas o curvas).

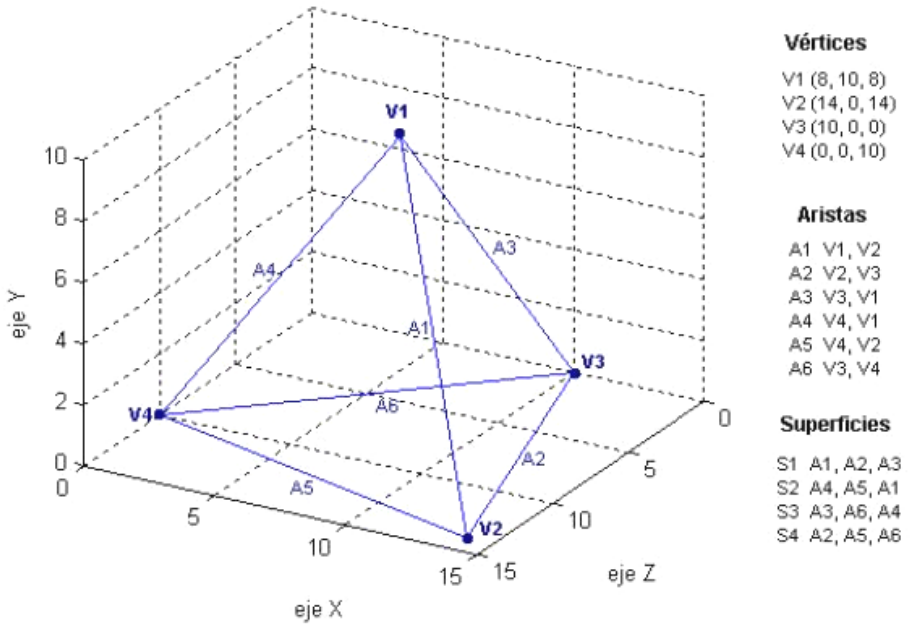


Figura 2 Representación de Frontera de un Tetraedro (Pirámide) (I7).

Dependiendo del número de vértices o aristas empleado se puede obtener una mejor aproximación del objeto desea, aunque se debe sacrificar la memoria del computador que realiza el modelado, ya que al insertar toda esta información, se puede obtener un incremento drástico de la información que ofrece el modelo como podemos apreciar a continuación:

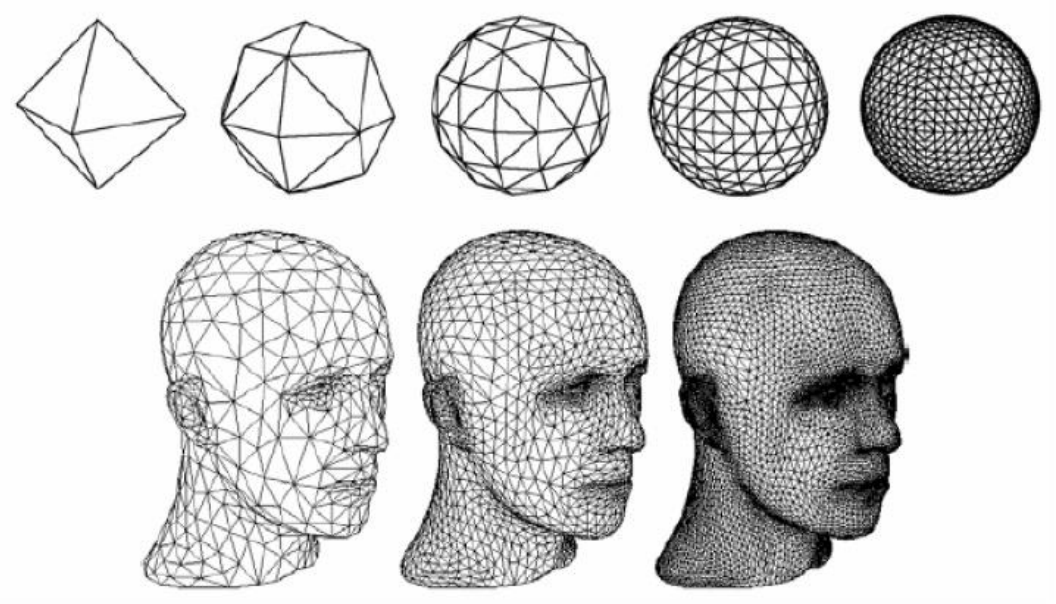


Figura 3 Representación de Frontera, Mallado de Precisión (I8).

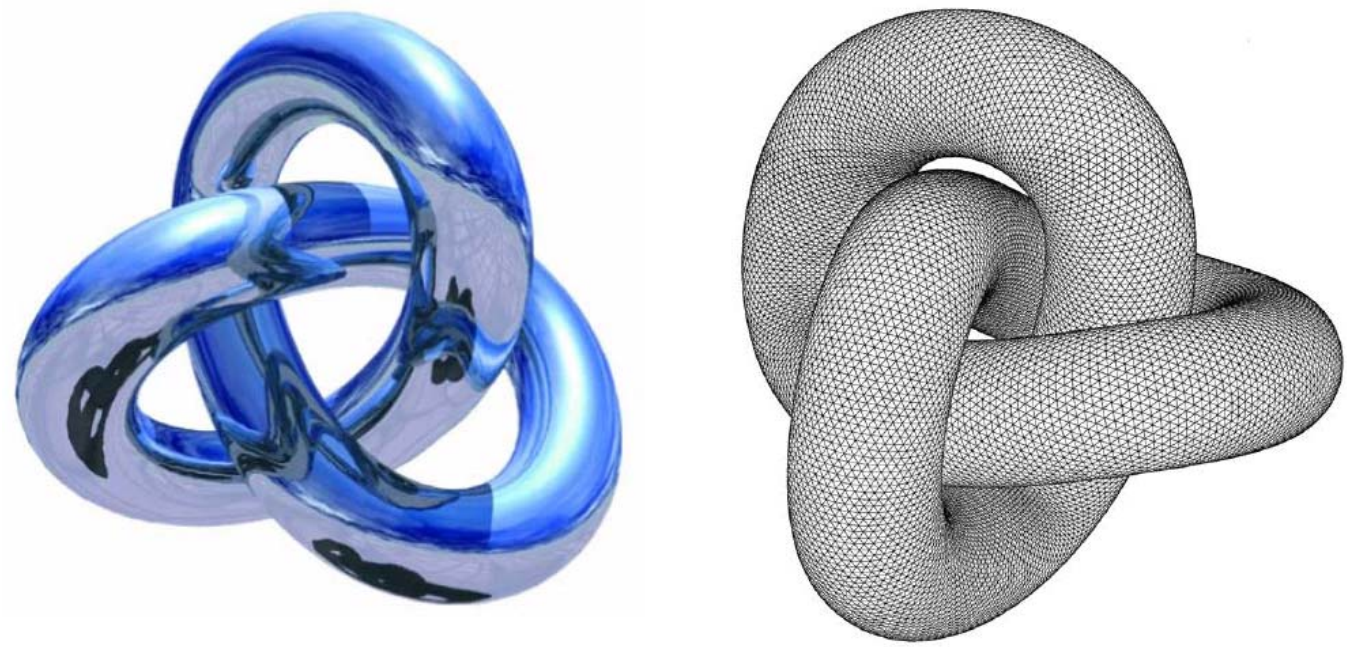


Figura 4 Aproximación de Superficies Mediante Mallas (I8)

Éste modelo será desarrollado con mayor detalle en el Capítulo IV del presente trabajo.

Representación mediante Árboles BSP:

Proviene de las siglas *Binary Space Partition* o en español árboles binarios de partición. En esta representación se dividen recursivamente el espacio en pares de sub espacios cada uno separado por un plano de orientación y posición arbitrario en el cual cada nodo interno del árbol BSP esta relacionado con un plano y tiene dos nodos hijos, cada uno para cada lado del plano, los mismos se clasifican en función de la normal de dicho plano como dentro / detrás para el hijo de la izquierda o delante / fuera para el de la derecha. Si el sub espacio generado se dividiera aún más, el hijo que lo formaría se convierte ahora en la raíz del sub espacio, si por el contrario no se continuase subdividiendo el espacio y éste es homogéneo, el nodo formado es una hoja y representaría completamente un sitio dentro o fuera del sólido como se muestra en la Figura representada a continuación.

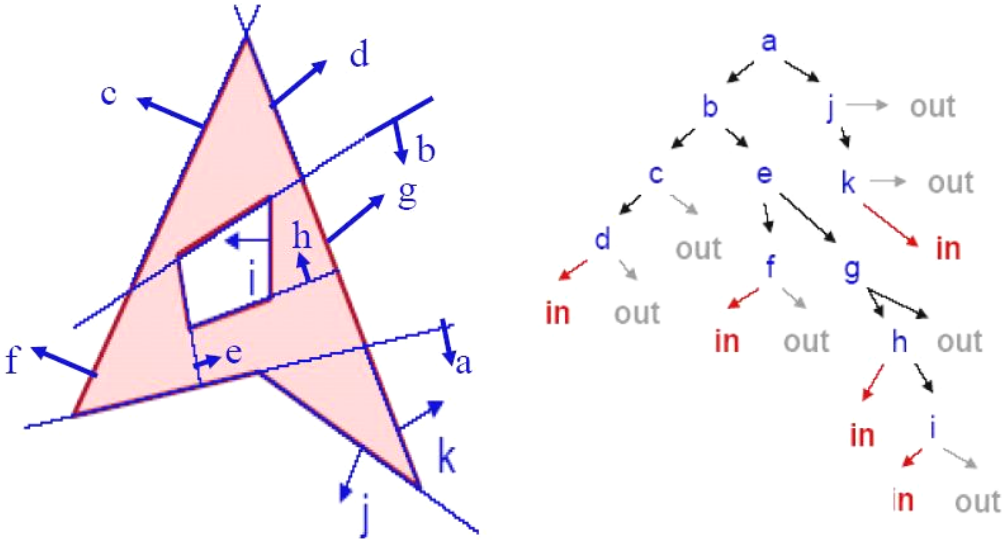


Figura 5 Representación Mediante BSP de una Figura (I8)

Representación mediante Geometría de Sólido Constructiva:

El método basado en el *Constructive Solid Geometric* o CSG, se basa mediante el uso de operadores boléanos y primitivas simples para la formación de sólidos más complejos, éste método se apoya en el árbol binario para la identificación de los nodos, por lo que las aristas deben estar ordenadas, en la figura 6 se presenta un ejemplo general de las operaciones requeridas para obtener el sólido descrito.

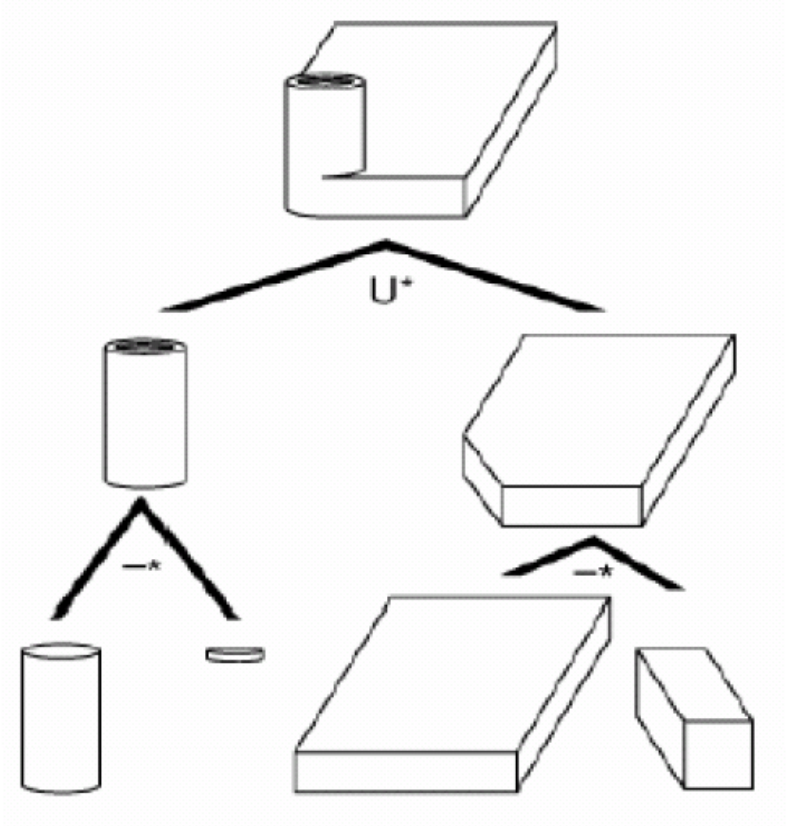


Figura 6 Representación mediante CSG de un sólido.[MANT88]

Representación mediante árboles octales:

Estos métodos de representación, incluyendo entre otros el *modelo del árbol octal*, son más bien usados en el campo del análisis numérico, las bases de los datos geográficos y el tratamiento de imágenes. La idea básica, según explica Marcheix [MARC96], consiste en representar los objetos a través de una subdivisión del espacio. En un primer tiempo, el espacio R3 que contiene al objeto es entonces subdividido con la ayuda de una rejilla tridimensional uniforme. En general esta rejilla es absolutamente regular, lo que conlleva a una discretización en elementos cúbicos del espacio inicial. En un segundo tiempo, los elementos que intersecan el interior del volumen son marcados, definiendo así una representación del objeto. La figura 7 da un ejemplo de subdivisión en elementos cúbicos, para el caso de un objeto relativamente simple.

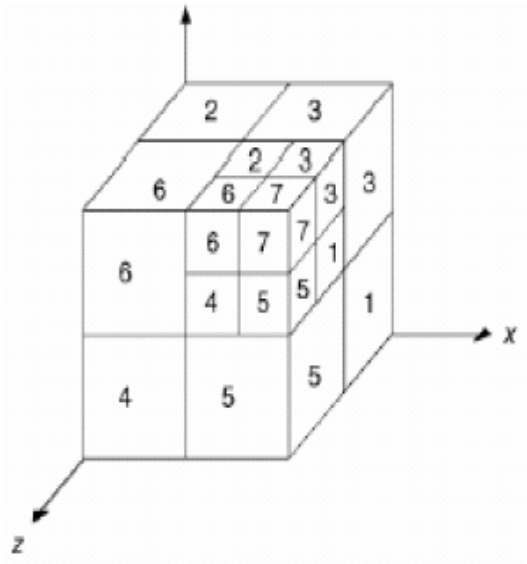


Figura 7 Representación Octal (I8).

Esta representación es muy costosa. También, con el fin de reducir la memoria necesaria, los *árboles octales* son frecuentemente usados para fusionar algunos elementos. Tomemos el ejemplo de una subdivisión cúbica. La idea consiste en reagrupar 8 cubos adyacentes que tengan el mismo tamaño en un solo cubo. Se obtiene de esta forma un cubo 8 veces más voluminoso. Este proceso se repite hasta que no existe ningún otro tipo de unión posible. (Figura 8.a). Finalmente, estas relaciones de adyacencia son almacenadas en una estructura de árbol octal, donde los nodos representan una agregación de los cubos que no han sido marcados, y las hojas

representan un cubo marcado, un cubo que no intersecan para nada el sólido, o un cubo donde se ha alcanzado el tamaño máximo de subdivisión (Figura 8.c).

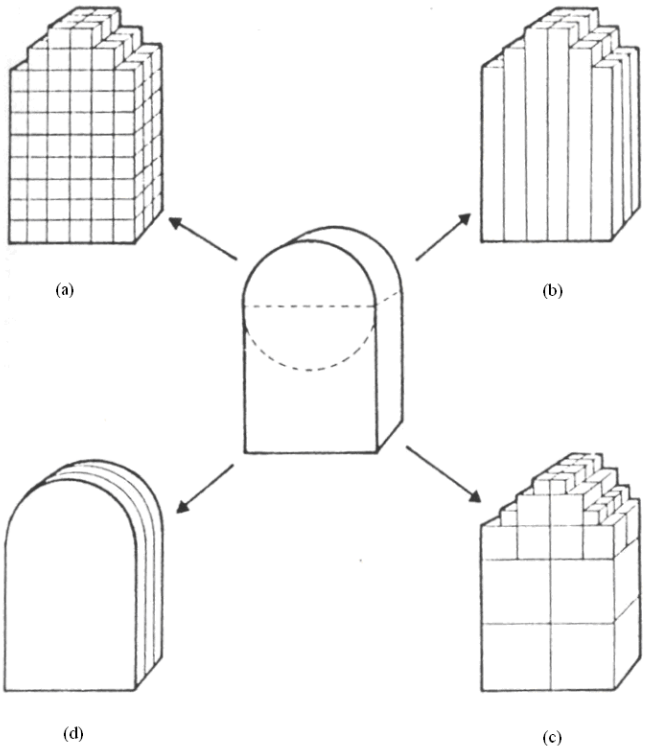


Figura 8 Representación Octal Desarrollo [MANT88]

Ahora si hablamos de una representación en 2 dimensiones (2D) nos estaremos refiriendo a un Árbol de Cuadrantes (Quadtree), que por definición aplica el mismo concepto que el Octal pero a dos planos: X e Y; de igual forma si 4 cuadrantes consecutivos están completamente llenos o completamente vacíos, se aplicaría una unificación de cuadrantes, esta representación se observa en la Figura 9.

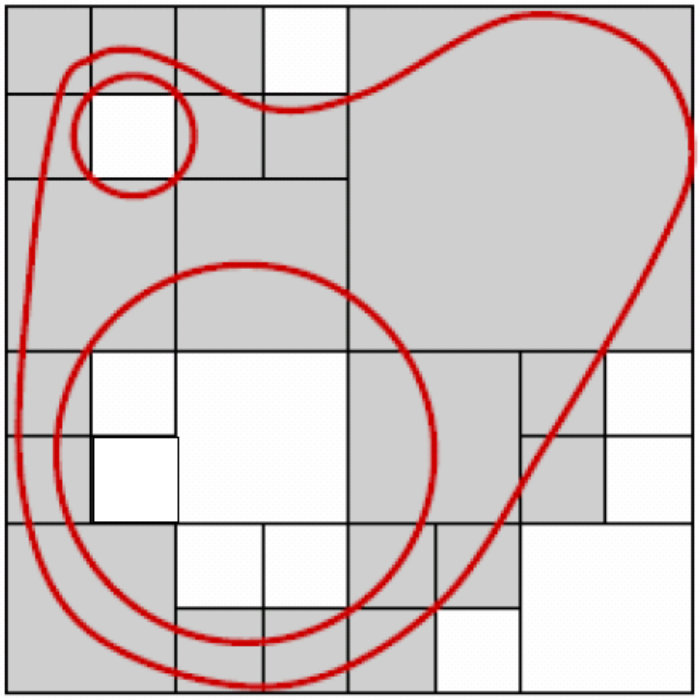


Figura 9 Representación de Cuadrantes de una figura 2D. (I8)

Precisión: Los métodos de partición de espacio y de representaciones poligonales de fronteras sólo producen aproximaciones a varios objetos. En algunas aplicaciones, como la determinación de una trayectoria para un robot, esta limitación no es una desventaja, siempre y cuando la aproximación se calcule con una resolución apropiada (por lo general no muy fina). Sin embargo, la resolución necesaria para producir gráficos visualmente atractivos o para calcular con precisión suficiente la interacción de objetos puede ser demasiado elevada para ser práctica. Las técnicas de sombreado suave, no corrigen los artefactos visuales ocasionados por las obvias aristas de los polígonos. Por lo tanto, los sistemas que permiten utilizar gráficos de alta calidad generalmente emplean CSG con primitivas no poliédricas y representaciones de frontera que permiten superficies curvas. La generación de ejemplares de primitivas también puede producir imágenes de alta calidad, pero no permite combinar dos objetos sencillos usando operadores de conjuntos booleanos

Dominio: El dominio de los objetos que se pueden representar con la generación de ejemplares de primitivas y los barridos es limitado. En cambio los métodos de partición espacial pueden representar cualquier sólido, aunque muchas veces sólo como una aproximación. Si se ofrecen otros tipos de caras y aristas además de los polígonos acotados por líneas rectas, es posible utilizar las representaciones de fronteras para representar una gama muy variada de objetos. Sin embargo, varios sistemas de representación de fronteras están restringidos a tipos de superficies y topologías sencillas. Por ejemplo, es probable que sólo puedan codificar combinaciones de cuádricas que sean 2-manifold.

Unicidad: Los únicos métodos que garantizan la unicidad de una representación son el árbol de octantes y la numeración de ocupación espacial, ya que ofrecen una sola forma de representar un objeto de tamaño y posición especifico. En el caso de los arboles octantes, hay que efectuar cierto procesamiento para asegurarse de que la representación se haya reducido totalmente (es decir, que ningún nodo gris tenga todos los hijos negros o blancos). La generación de ejemplares de primitivas por lo general no garantiza la unicidad; por ejemplo, una esfera se puede representar con una primitiva esférica o elíptica. Sin embargo, la unicidad se puede asegurar si se elige con cuidado el conjunto de primitivas.

Validez: De todas las representaciones, las de frontera son las que se distinguen por ser las más difíciles de validar. No sólo es posible que las estructuras de datos vértices, caras y aristas sean inconsistentes, sino que también puede haber caras o aristas que se intersequen. En cambio, cualquier árbol BSP representa un conjunto espacial válido, aunque no necesariamente un sólido acotado. Sólo hay que efectuar una sencilla revisión sintáctica local para validar un árbol CSG (que siempre está acotado si sus

primitivas están acotadas) o un árbol de octantes, y no se requiere verificación para la numeración de ocupación espacial.

Cerradura: Las primitivas creadas utilizando la generación de ejemplares de primitivas no se pueden combinar y los barridos simples no son cerrados al aplicarse operaciones booleanas. Por lo tanto, ninguno de estos métodos se emplea como representación interna en sistemas de modelado. Aunque ciertas representaciones de fronteras pueden presentar problemas de cerradura con operaciones booleanas (por ejemplo la incapacidad de representar las que no sean 2-manifold), estos casos generalmente se pueden evitar.

Densidad y eficiencia: Los métodos de representación generalmente se clasifican de acuerdo con su capacidad para producir modelos *evaluados* o *no evaluados*. Los modelos *no evaluados* contienen información que debe procesarse (o evaluarse) para efectuar operaciones básicas, como la determinación de la frontera del objeto. En lo que respecta a la utilización de operaciones booleanas, la CSG crea modelos no evaluados, ya que cada vez que se realizan cálculos hay que recorrer el árbol y evaluar las expresiones. Por lo tanto, las ventajas de la CSG son lo compacto de la representación y su capacidad para registrar rápidamente operaciones booleanas y cambios de transformaciones, así como deshacer éstas con rapidez, ya que sólo hay que construir nodos del árbol. Los árboles de octantes y BSP también se pueden considerar como modelos no evaluados, al igual que una secuencia de Operadores de Euler que crea una representación de frontera. Por otra parte, las representaciones de fronteras y la numeración de ocupación espacial a menudo se consideran modelos *evaluados* en lo que se refiere a que ya se han realizado varias de las operaciones booleanas utilizadas para crear un objeto. Observe que la utilización de éstos términos es relativa; por ejemplo, si la operación que se realizará consiste en la determinación de si un punto está dentro de un objeto, se puede llevar a cabo más trabajo al evaluar una representación de frontera que al evaluar un árbol CSG equivalente.

El desarrollo de la interfaz con el usuario de un sistema de modelado de sólidos nos ofrece una oportunidad excelente para poner en práctica las técnicas de diseño. Hay diversas técnicas que se representan para las interfases gráficas, incluyendo la aplicación directa de operadores de Euler. En los sistemas CSG se puede permitir que el usuario edite el objeto modificando o reemplazando uno de los subárboles o sólidos de hoja. Podemos definir operaciones de fusión y biselado para suavizar la transición de una superficie a otra. Las interfases con el usuario de los sistemas que han tenido éxito son muy independientes de la representación interna que se elija. Sin embargo, la excepción es la generación de ejemplares de primitivas, ya que alienta a los usuarios a pensar en los objetos en función de parámetros de propósito especial.

Existen varias formas equivalentes de describir la misma curva. Por ejemplo, la Interfaz entre el usuario y un sistema de dibujo de curvas puede permitir que el usuario registre curvas controlando los vectores tangente hermitianos o especificando puntos de control de Bézier. En forma similar, un sistema de modelado de sólidos, puede permitir que el usuario cree objetos en función de varias representaciones y los almacene con otra. Como sucede con las representaciones de curvas, cada representación de entrada puede tener ventajas que hagan de ella la opción más natural para crear el objeto.

La precisión con que deben especificarse los objetos muchas veces establece que deben ofrecerse medios para determinar mediciones en forma precisa, por ejemplo a través del dispositivo localizador o con una entrada numérica. Puesto que la posición de un objeto muchas veces depende de la de otros, las interfases con frecuencia permiten restringir un objeto con respecto a otro. Una técnica relacionada consiste en proporcionar al usuario la capacidad para definir líneas de malla a fin de restringir la posición de los objetos.

Algunos de los problemas fundamentales del diseño de una interfaz ara modelado de sólidos son los ocasionados por la necesidad de manipular y presentar objetos tridimensionales usando dispositivos de interacción y pantallas bidimensionales. Muchos sistemas tratan algunos de estos problemas ofreciendo varias ventanas de presentación, que permiten al usuario ver el objeto simultáneamente desde posiciones diferentes.

CAPÍTULO III

OPERACIONES BOOLEANAS

Mediante la implementación de operaciones Booleanas se permite obtener un objeto a partir de la operación (suma, resta o intersección) de 2 o más primitivas, La aplicación se basa a su vez en los siguientes conceptos:

Entorno de un vértice:

El entorno de un vértice es el conjunto de vértices, caras y aristas que están conectados a él, ordenados de forma radial.

Sector:

La parte de una cara, inmediatamente adyacente al vértice (ver partes sombreadas de la figura)

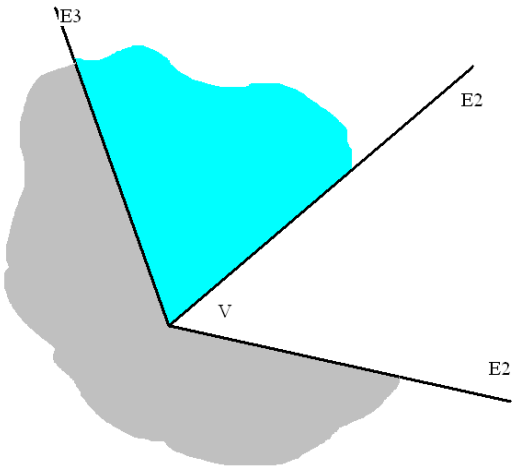


Figura 10. Entidades topológicas alrededor de un vértice V

Clasificación:

Se determinará una clasificación genérica, basada en la clasificación DENTRO / SOBRE / FUERA de un sólido A respecto a otro sólido B, todo esto se hará con el vértice (dimensión 0) como elemento básico, para entidades de dimensionalidad mayor se establece la siguiente clasificación:

Arista (1D): En base a la clasificación de sus vértices extremos.

Caras (2D): en base a la clasificación de las aristas que la delimitan.

Es importante recalcar que se debe asignar también una jerarquía a la hora de desarrollar los modelos, por ello, se define en función de las operaciones del siguiente modo:

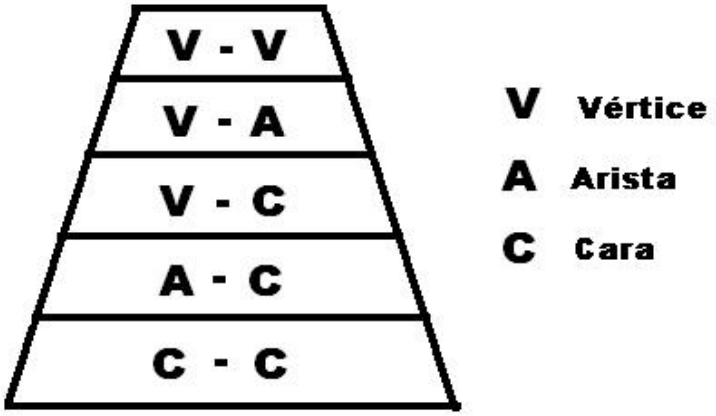


Figura 11. Esquema de Jerarquías de Operaciones

Clasificación del entorno de un vértice

Utilizando la clasificación de un vértice, en combinación con los que lo rodean, se puede identificar los componentes comunes de los sólidos que se intersecan, según la forma de compartir dichos elementos

Clasificación de Contorno

La operación de Conjunto puede ser considerada como la división de un poliedro A según la referencia del poliedro B, como se muestra a continuación en la figura

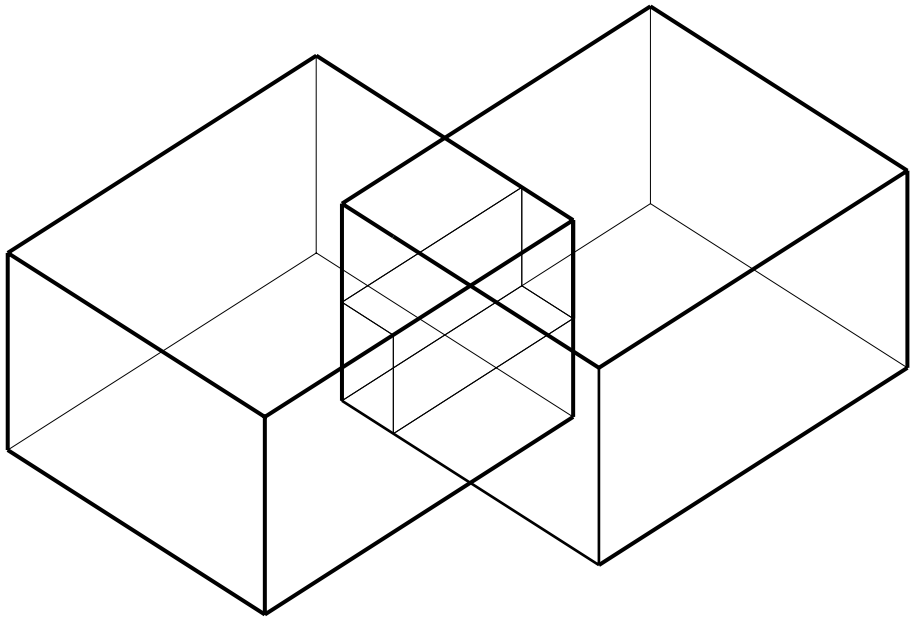


Figura 12. Operación de conjunto entre A y B

Si tomáramos la superficie de A como b(A) y la superficie de B como b(B), entonces los objetos resultantes de una operación general de partición pueden ser llamados:

1. A ins B : la parte de b(A) que esta dentro de B
2. A out B: la parte de b(A) que está fuera de B
3. B ins A: la parte de b(B) que esta dentro de A
4. B out A: la parte de b(B) que está fuera de A

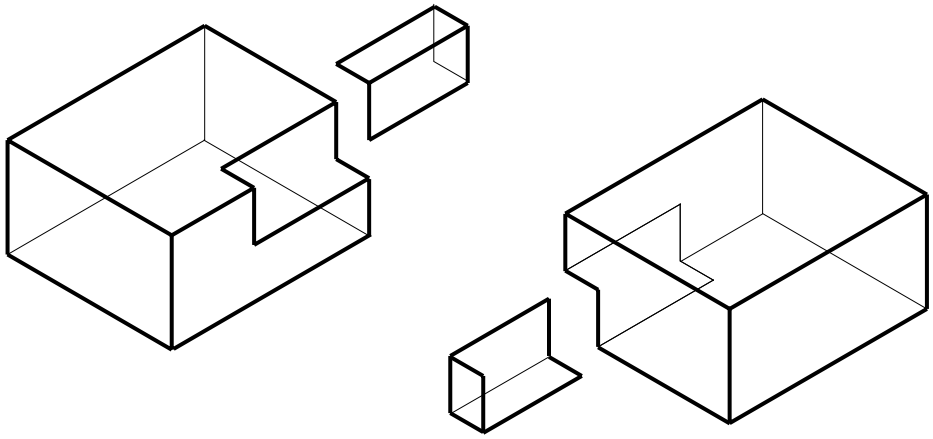


Figura 13. Intersección de sólidos.

Componentes de la clasificación de Contornos:

Con estos cuatro objetos podemos entonces denominar la clasificación de los contornos de A y B, a partir de la cual se pueden calcular las combinaciones booleanas:

$A \cup B = AoutB \oplus BoutA$ Unión

$A \cap B = AinsB \oplus BinsA$ Intersección

$A \; D \; B = AoutB \oplus (BinsA)^{-1}$ Diferencia

Entonces de forma esquemática, quedaría representada la solución de la siguiente manera:

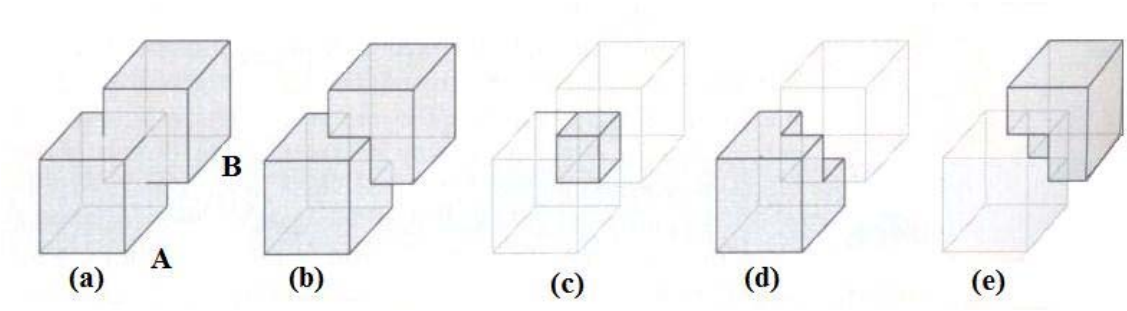


Figura 14 Visualización Intersección de Sólidos [FOLE94]

La operación de pegado (Gluing, [MANT88]) es repesentada por ⊕, mientras que : $(BinsA)^{-1}$ representa el "complemento" de B ins A, es decir BinsA después de invertir la orientación de todas sus caras.

Surge entonces la necesidad de resolver las ambigüedades que aparecen a lo largo de las entidades coincidentes, como es el caso de caras coplanares, entonces se complementa la clasificación de 4 con una clasificación de 8 vía de los sólidos A y B, que añade los componentes A on B^+ , A on B^-, B on A^+ y B on A^-. el primero consiste de la partes de b(A) que están sobre b(B) de tal manera que las normales a las caras correspondientes están en la misma dirección, mientras que A on B^- consiste en las partes que se solapan, donde las normales tienen direcciones opuestas. Luego las subsecuentes se definen de igual manera.

A partir de estos componentes (de 8 vías), el resultado de una operación de conjunto puede ser calculado de la siguiente manera:

$$A \cup B = AoutB \oplus BoutA \otimes AonB^{+}$$

$$A \cap B = AinsB \oplus BinsA \otimes AonB^{+}$$

$$A \; D \; B = AoutB \oplus (BinsA)^{-1} \otimes AonB^{-}$$

Simplificando los cálculos, los componentes "sobre" de la clasificación de 8 vías al combinarse con los de la clasificación de 4 vías, permite obtener un conjunto de reglas de reclasificación, como lo son:

B on A^{-}	Operación	A on B^{+}	A on B^{-}	B on A^{+}
B ins A	Unión	A out B	A ins B	Bins A
B out A	Intersección	A ins B	A out B	B out A
B out A	Diferencia	A ins B	A out B	B out A

La primera fila indica que para la operación de unión A on B- debería ser tratado como Aout B y A on B- como parte de A insB. BonA- y BonA+ son considerados ambos como BinA. Por lo tanto solamente AonB aparecerá en el resultado final.

Independientemente de cómo representemos los objetos, nos gustaría poder combinarlos para formar nuevos. Uno de los métodos más intuitivos y comunes para combinar objetos son las **operaciones booleanas de conjuntos** (como se explicó anteriormente) Unión, Intersección y diferencia. Estas operaciones son los

equivalentes tridimensionales de las operaciones booleanas bidimensionales familiares. Sin embargo, la aplicación de una operación booleana ordinaria de conjuntos a dos objetos sólidos no necesariamente produce un objeto sólido. Por ejemplo, la intersección ordinaria de dos cubos que se unen en un solo vértice es un punto.

En lugar de emplear los operadores booleanos ordinarios de conjuntos, se deben emplear **operadores regularizados de conjuntos,** definidos de manera que las operaciones con sólidos siempre generen sólidos, Por ejemplo, la Intersección booleana regularizada de dos cubos que se unen en un solo vértice es el objeto nulo.

Para examinar la diferencia entre los operadores ordinarios y los regularizados, podemos considerar cualquier objeto como definido por un conjunto de puntos y dividido en puntos interiores y puntos de frontera.

Los puntos de frontera son aquellos cuya distancia al objeto y al complemento del objeto es **cero**. Los puntos de frontera no tienen que formar parte del objeto. Un conjunto cerrado contiene todos sus puntos de frontera, mientras que un conjunto abierto no contiene ninguno. La unión de un conjunto, con el conjunto de sus puntos de frontera se conoce como cerradura del conjunto, que es en sí un conjunto cerrado. La frontera de un conjunto cerrado es el conjunto de sus puntos de frontera, mientras que el interior, consiste en todos los demás puntos del conjunto y por ende es el complemento de la frontera con respecto al objeto. La **regularización** de un conjunto esta entonces definida como la cerradura de de los puntos interiores del conjunto. Un conjunto que es igual a su propia regularización se conoce como **conjunto regular.** Un conjunto regular no puede contener ningún punto de frontera que no sea adyacente a un punto interior; es decir, no puede tener puntos, líneas o superficies de frontera “colgantes”. Luego entonces cada operador regularizado esta definido por su

operador ordinario. En conclusión, los operadores booleanos regularizados de conjuntos, producen sólo conjuntos regulares cuando se aplican a conjuntos regulares.

Para explicar de mejor forma lo anterior expuesto, nos proponemos el ejemplo a continuación:

Sea A y B dos figuras como se muestra a continuación (a), y se quiere conocer el resultado de de la intersección (b) de ambos (A∩B) de la forma presentada,

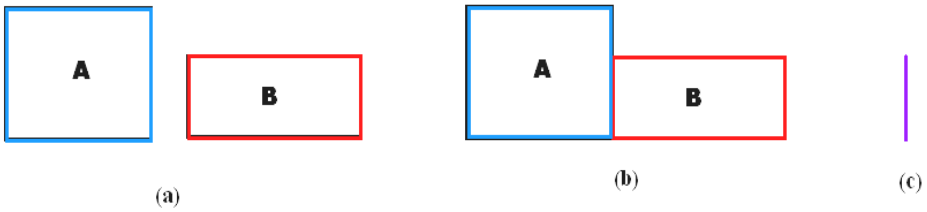


Figura 15. Operación de conjuntos booleanos regulares.

El resultado será el lado común de ellas (c), presentado en color morado, como la arista en común de los 2 elementos (rojo y azul); si observamos con detalle tenemos de la operación de 2 elementos 2-D el resultado de un elemento en 1D, esta operación es sólo permitida por Operadores Regularizados.

A continuación se plantean los ejemplos que describen la operación booleanas, paso a paso, la misma esta basada en la filosofía de Mäntylä [MANT88] acerca de la formación de nodos e intersección de aristas (la creación de nodos padres, hijos, etc.)

Dos objetos como se muestran en la figura 16, se desea unirlos, ellos al sobreponerse uno encima del otro se fijan lo que serían las uniones de caras, intersecciones de aristas con caras y vértices dentro del sólido opuesto, esta región se muestra de color blanco en (b), dependiendo de la operación booleanas que se desarrolla, las intersecciones generan los nodos padres y los nodos hijos (clasificación), es entonces que el programa excluye del gráfico los nodos ya sea los padres o los nodos hijos; en el caso de la interseccion se excluyen los nodos padres que son externos a la subregión formada, posteriormente se construye las nuevas aristas que generarán el nuevo sub espacio, creando una nueva lista de vértices, nodos y caras (c) y por último con la nueva lista de nodos, caras y vértices queda definido el nuevo sólido (d).

Intersección:

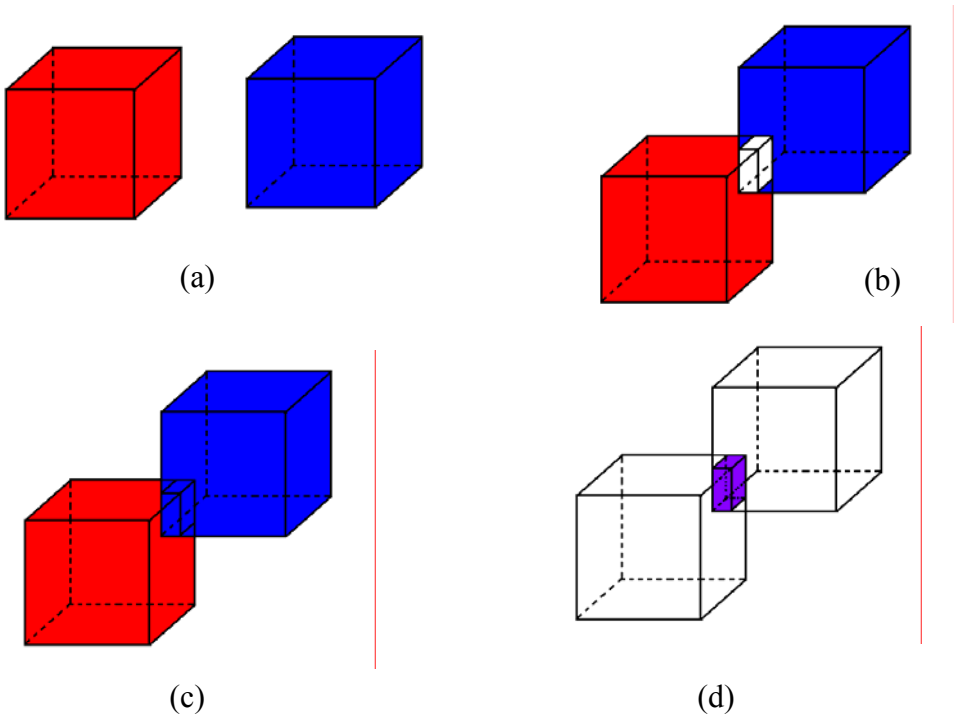


Figura 16. Intersección de dos Sólidos (paso a paso).

Para el caso de una unión (figura 17) el proceso es muy similar al anterior, se muestran los dos sólidos que van a formar el sólido resultante (a), luego estos se superponen y se determinan las intersecciones de vértices, caras y aristas (b) representadas como el sub espacio blanco. Posteriormente según la clasificación de vértices, caras y nodos (nodos padres y nodos hijos) el proceso de unión toma únicamente los nodos padres externos a dicho espacio y los nodos hijos los renombra nodos principales para una posterior operación (c). De igual manera existe un reordenamiento en la lista de nodos, vértices y caras para definir el nuevo sólido.

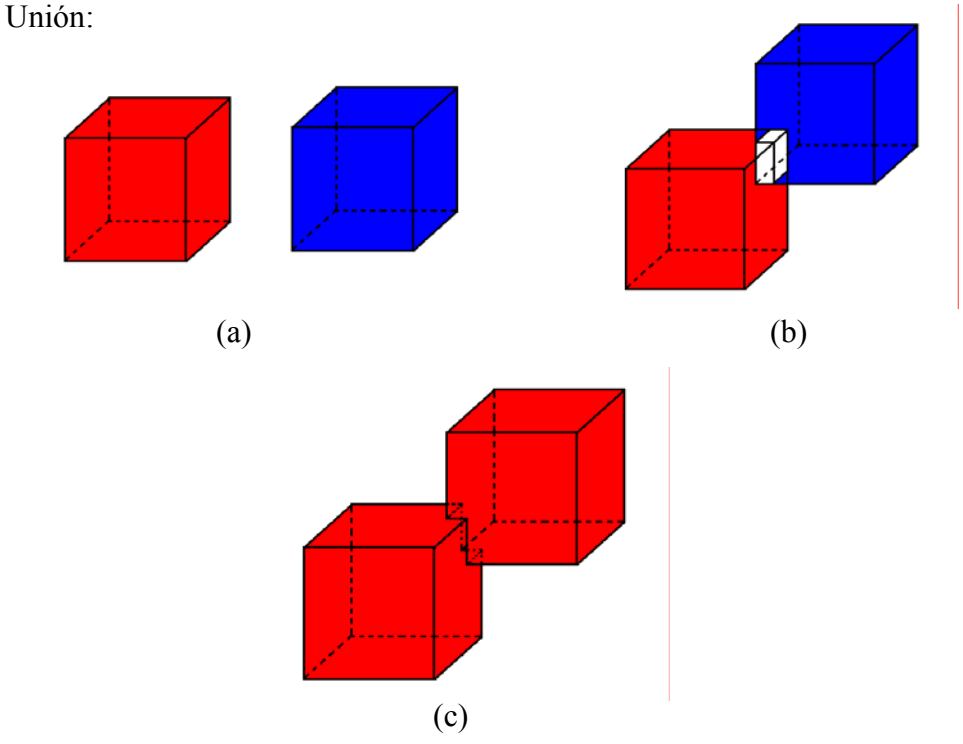


Figura 17. Unión de dos Sólidos (paso a paso).

Por último, al desarrollar una operación de diferencia de sólidos (figura 18), se muestra paso a paso el proceso (muy similar a los anteriores) iniciando con los dos sólidos (a) superponiendo los mismos (b) para poder hallar la intersección, luego clasificando y formando los nodos (padres e hijos) como se aprecia en (c) sólo que al final en (d) si se tomó como operación de diferencia el cuadro rojo menos el cuadro azul, serán los nodos padres rojos los que definan mi nuevo sólido y los nodos hijos rojos los que formen las nuevas caras, el resto de los nodos desaparece y queda definido mi sólido respuesta a la operación (e).

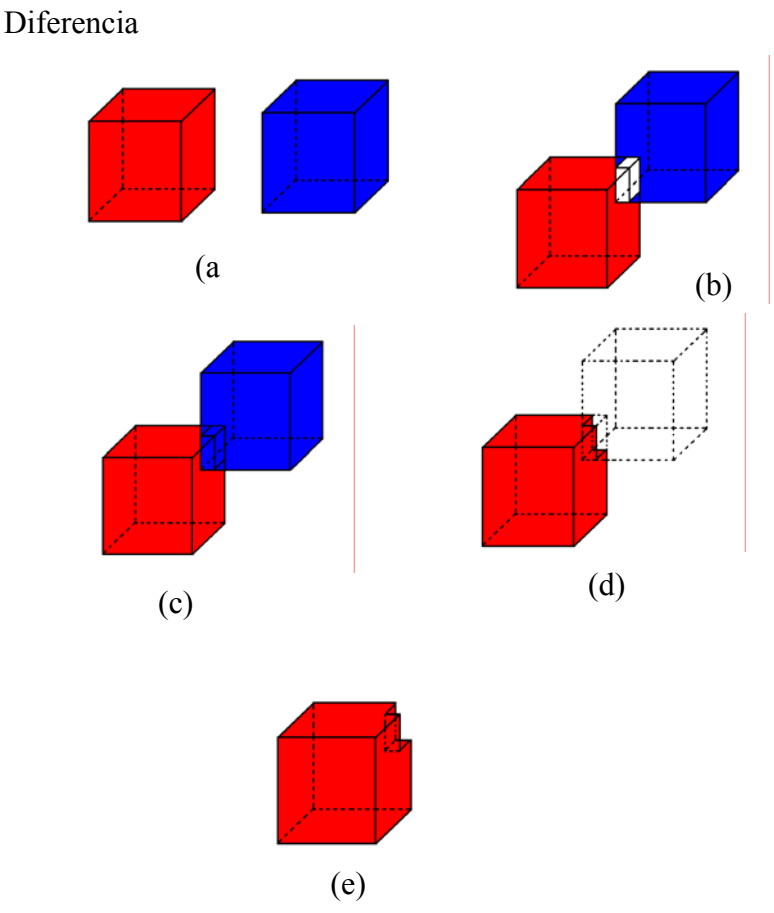


Figura 18. Diferencia entre dos Sólidos (paso a paso).

CAPÍTULO IV

ESTRUCTURA DE DATOS DE FRONTERA

Los constituyentes básicos de los modelos de frontera son los tipos de objetos *caras, aristas, y vértices,* y la información geométrica unida a ellos. Adicionalmente a la información geométrica tal como las ecuaciones de las caras, de las curvas, y las coordenadas de los vértices, un modelo de frontera debe también representar como las caras, aristas, y vértices están relacionadas unas con otras.

Todos los modelos de frontera representan caras en términos de nodos explícitos de una estructura de datos. Según Mäntylä [MANT88], existen entonces varias alternativas para la representación geométrica y topológica de un modelo de este tipo.

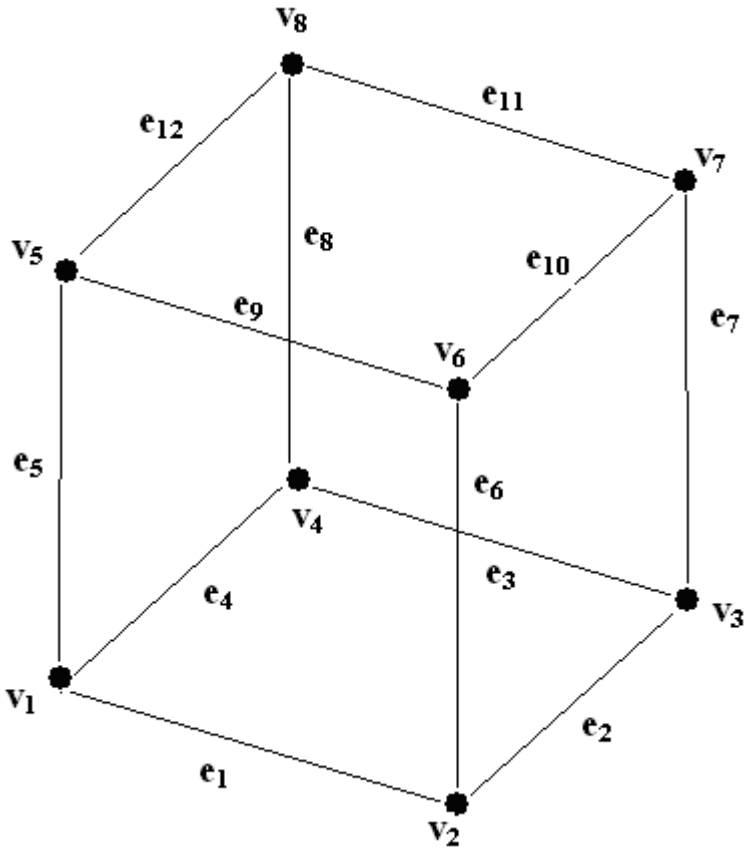
Modelo de frontera basado en aristas:

Este modelo representa una cara frontera en términos de una secuencia cerrada de aristas, lo que se denomina simplemente *lazo.* Los vértices de las caras son representados solamente a través de aristas.

En la figura 19 se muestra la estructura de datos de este modelo para un simple cubo.

Dicha estructura indica una orientación para cada arista; es decir, se considera que la arista e_1 está orientada positivamente desde el vértice v_1 al vértice v_2. Las caras están orientadas en sentido horario, visto desde afuera del sólido.

Nótese que cada arista ocurre en exactamente dos caras, una en su orientación positiva, y la otra en su orientación negativa.



arista	***vértices***
e_1	v_1 v_2
e_2	v_2 v_3
e_3	v_3 v_4
e_4	v_4 v_1
e_5	v_1 v_5
e_6	v_2 v_6
e_7	v_3 v_7
e_8	v_4 v_8
e_9	v_5 v_6
e_{10}	v_6 v_7
e_{11}	v_7 v_8
e_{12}	v_8 v_5

vértice	***coordenadas***
v_1	x_1 y_1 z_1
v_2	x_2 y_2 z_2
v_3	x_3 y_3 z_3
v_4	x_4 y_4 z_4
v_5	x_5 y_5 z_5
v_6	x_6 y_6 z_6
v_7	x_7 y_7 z_7
v_8	x_8 y_8 z_8

cara	***arista***
f_1	e_1 e_2 e_3 e_4
f_2	e_9 e_6 e_1 e_5
f_3	e_{10} e_7 e_2 e_6
f_4	e_{11} e_8 e_3 e_7
f_5	e_{12} e_5 e_4 e_8
f_6	e_{12} e_{11} e_{10} e_9

Figura 19. Estructura de datos para un modelo basado en aristas
(Según Mäntylä [MANT88])

Modelo de frontera basado en estructura de datos de arista con alas:

Esta estructura de datos (ver figura 20) es más avanzada que la estructura basada en aristas, según lo expuesto por Mäntylä [MANT88], puesto que también incluye información explícita de cada objeto básico (caras, aristas, vértices) a través de nodos relacionados entre sí, que permiten determinar, por ejemplo, cual arista perteneciente a un lazo de una cara le sigue a otra arista, o cual es la arista referencial de una cara, es decir, cual es la arista que inicia el lazo de una cara.

arista	*v.inicial*	*v.final*	*sig_hor*	*sig_anti_hor*
e_1	v_1	v_2	e_2	e_5
e_2	v_2	v_3	e_3	e_6
e_3	v_3	v_4	e_4	e_7
e_4	v_4	v_1	e_1	e_8
e_5	v_1	v_5	e_9	e_4
e_6	v_2	v_6	e_{10}	e_1
e_7	v_3	v_7	e_{11}	e_2
e_8	v_4	v_8	e_{12}	e_3
e_9	v_5	v_6	e_6	E_{12}
e_{10}	v_6	v_7	e_7	e_9
e_{11}	v_7	v_8	e_8	e_{10}
e_{12}	v_8	v_5	e_5	E_{11}

vértice	*coordenadas*
v_1	$x_1\ y_1\ z_1$
v_2	$x_2\ y_2\ z_2$
v_3	$x_3\ y_3\ z_3$
v_4	$x_4\ y_4\ z_4$
v_5	$x_5\ y_5\ z_5$
v_6	$x_6\ y_6\ z_6$
v_7	$x_7\ y_7\ z_7$
v_8	$x_8\ y_8\ z_8$

Cara	*prim_arista*	*Signo*
f_1	e_1	+
f_2	e_9	+
f_3	e_6	+
f_4	e_7	+
f_5	e_{12}	+
f_6	e_9	-

Figura 20. Estructura de datos de arista con alas (Según Mäntylä [MANT88])

De igual forma se puede saber, a partir de una arista (la cual pertenece a dos caras simultáneamente), cual es arista que le sigue recorriendo una de las caras en sentido horario, o cual es la arista que le sigue haciendo un recorrido anti horario de la otra cara.

Por otra parte, las caras solamente necesitan la inclusión de una arista arbitraria inicial, y de un signo (+ o -) que acompañe a la arista, de tal forma que dependiendo del signo, se sepa qué cara se está identificando.

Estructura de arista radial:

El modelado por superficie, permite modelar los objetos non-manifold y los manifold de tal forma que se extiende el dominio para dicha representación permitiendo entonces trabajar con hilos, superficies, sólidos y modelado celular simultáneamente en el mismo ambiente.

La Estructura de arista radial es la primera representación de modelado de elementos Non – manifold que explícitamente representa las adyacencias topológicas .Las adyacencias entre vértices, aristas, caras y volúmenes, se representan proveyendo una plantilla con una variedad de curvas representadas en un sistema de modelado geométrico.

La importancia de la manipulación de objetos non – manifold recae en las operaciones Booleanas de contorno, así como el análisis por métodos de elementos finitos que se pueden ejecutar en la misma representación.

El modelado geométrico de los elementos Non – Manifold haciendo uso de la representación por superficies permite como dominio

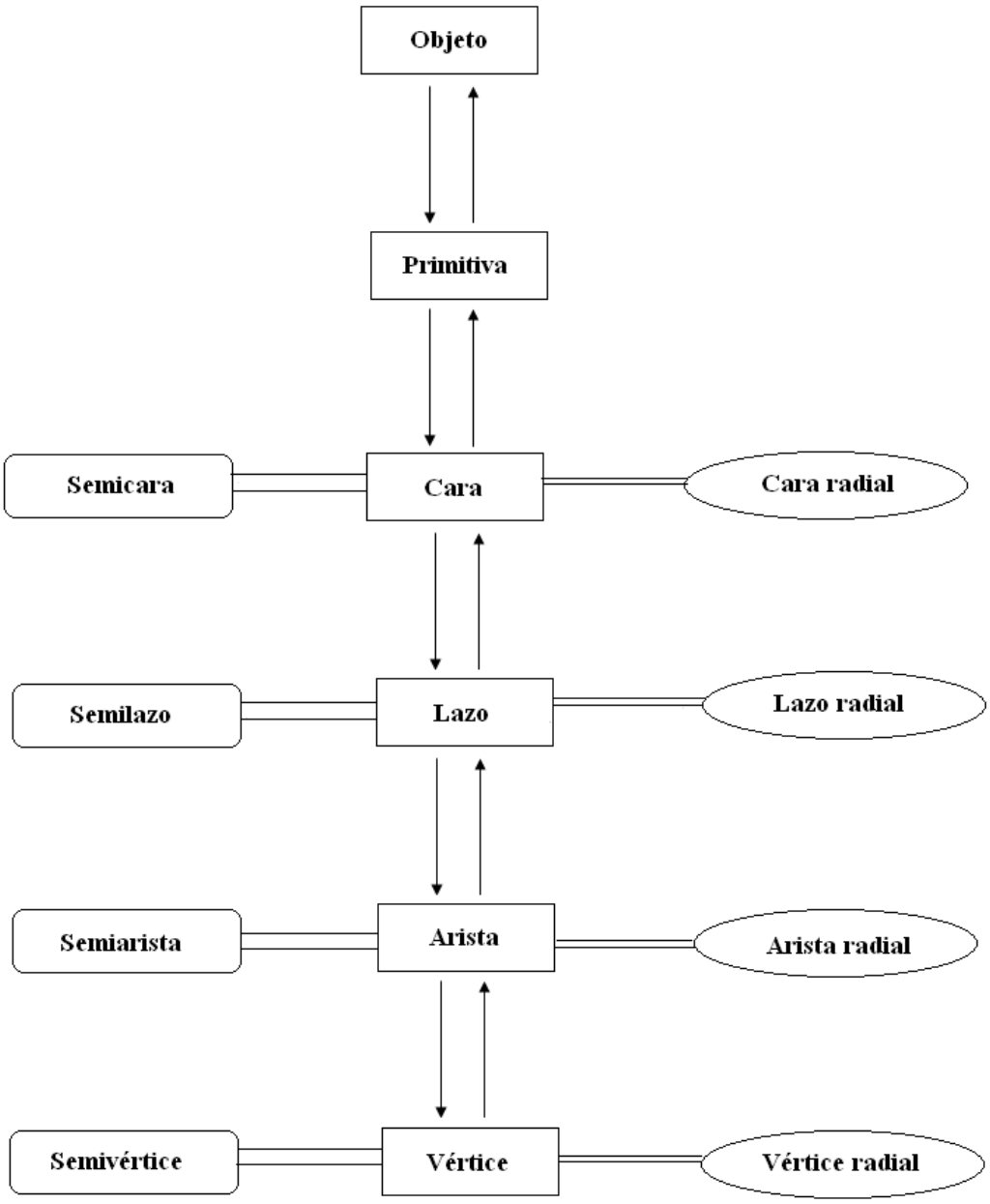
Modelo de frontera basado en estructura de datos de arista radial:

La estructura de datos de arista radial, desarrollada por Marchéix [MARC01], y utilizada para la implementación del modelo radial, es presentada bajo la forma de un gráfico de incidencia descrito en la figura

.

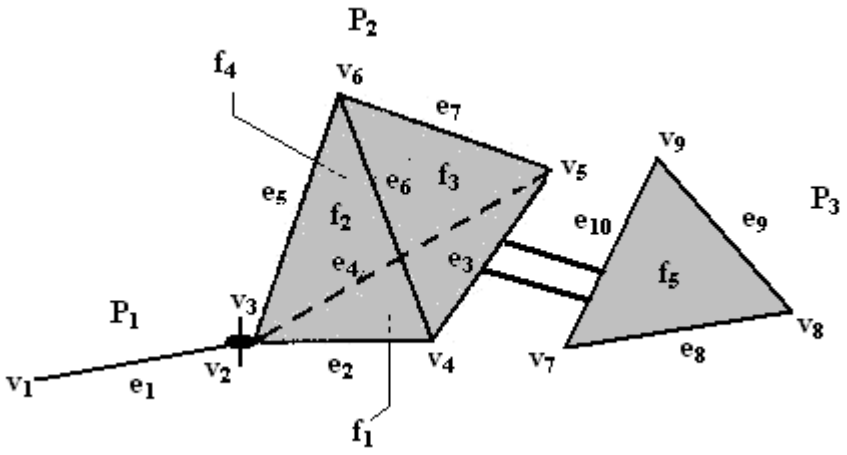
El objeto se compone de tres primitivas (figura b): un hilo de alambre P_1, un tetraedro P_2, y una cara colgante P_3. Las condiciones de no variedad (non-manifold) son codificadas con uniones radiales a nivel del vértice V_{rad}, para V_2 y V_3, y al nivel de arista con e_{rad}, para e_3 y e_{10}.

En la estructura de datos de arista radial, se introducen nuevas nociones como la de semivértice, semiarista, y semicara. Una cara colgante, por ejemplo, poseerá dos orientaciones y en consecuencia dos semicaras, mientras que una cara que pertenezca a una primitiva volumétrica será constituida únicamente por una semicara que define el interior de la primitiva. De manera análoga se introduce los elementos *loop*, *loop radial* y *semiloop* para facilitar la manipulación de aristas adyacentes.



(a)

Figura 21: Estructura de datos del modelo radial
(Según Marchéix [MARC96])



(b)

Figura 22. Estructura de datos del modelo radial (Continuación)

La figura 23 muestra como se intersecan múltiples caras en una arista radial.

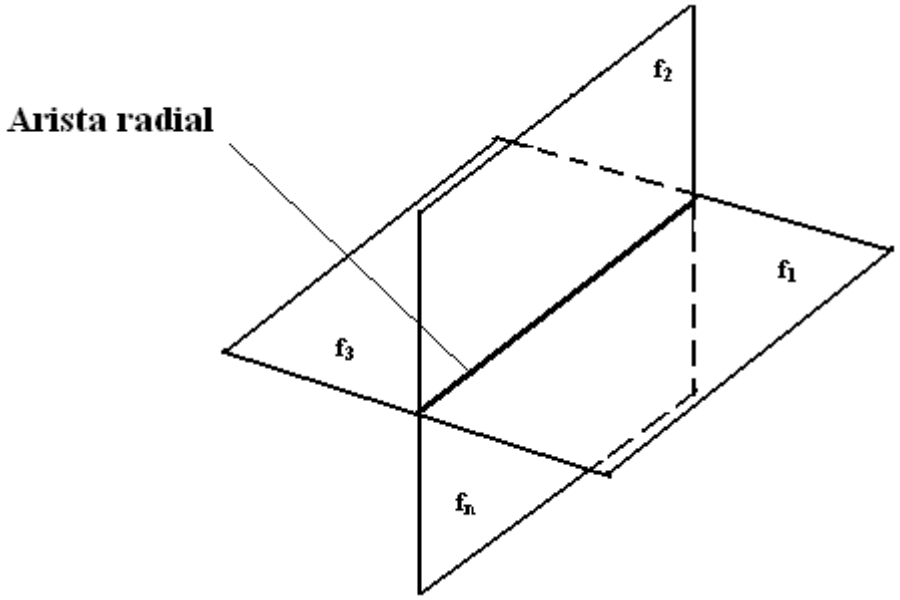


Figura 23. Múltiples caras se intersecan en una arista radial
(Según Marchéix [MARC96]).

CAPTÍTULO V

DESARAROLLO COMPUTACIONAL

Para poder analizar con detalle un programa es necesario crear una estructura entendible de todos y cada uno de los procesos que juegan un papel dentro de la metodología desarrollada.

A continuación se presenta un esquema que permite ver el funcionamiento, el corazón del programa, dicha metodología está basada en la filosofía de Mäntylä [MANT88]

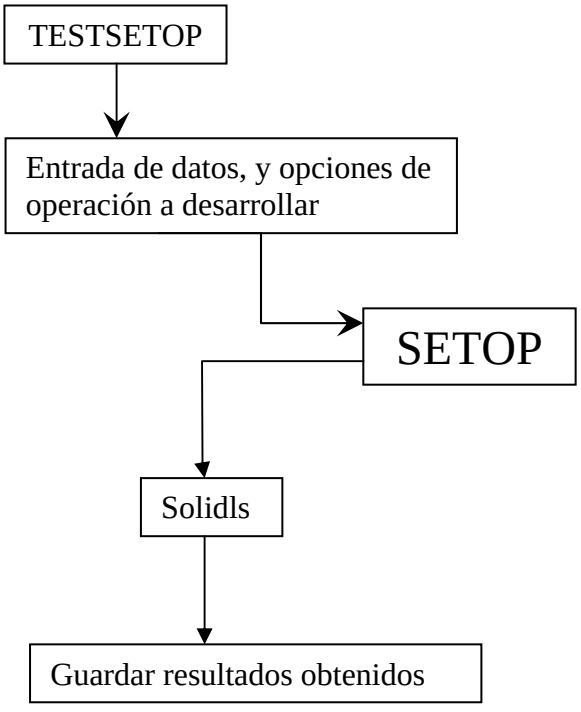


Figura 24 Árbol de funciones Testsetop.c (RESUMEN)

El director (main()) del programa se encuentra en TestSetop.c y es el que va a definir el orden en que se presentan o aplican las funciones que desarrollaran el propósito del programa, el esquema mostrado como SETOP es el corazón del Módulo de Operaciones Booleanas y a continuación se muestra el árbol de funciones de dicho proceso.

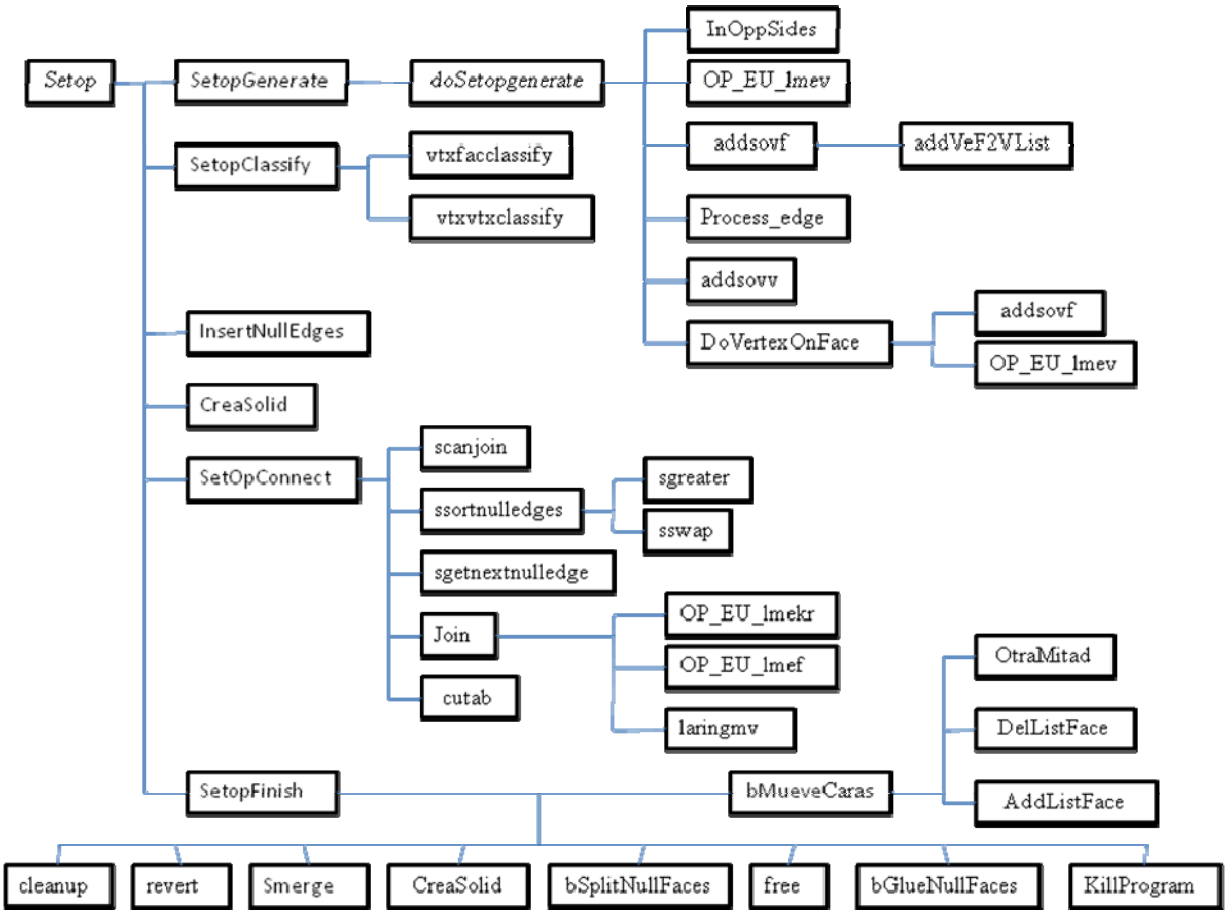


Figura 25 Árbol de funciones Setop.c (RESUMEN).

La función `Setop` es la encargada de realizar las operaciones de Sólidos, entiéndase las operaciones Booleanas de: Unión, Intersección y Diferencia. A continuación se explicará las funciones de interés del árbol de funciones, estas funciones cumplen un papel importante en lo que el proceso implementado se refiere.

`Addsovf:` Está encargada de identificar las intersecciones de las aristas del sólido B con las caras del sólido A, esto lo hace mediante la rutina `AddVeF2list` y es la que crea los nodos hijos en los casos de haber intersecciones, es importante la acotación de la variable `MAXVONF` que quiere decir el máximo número de vértices que puede haber en una cara, este parámetro permite limitar el recorrido que hará la función (función que limita "`CheckBounds`") para la generación de nodos hijos.

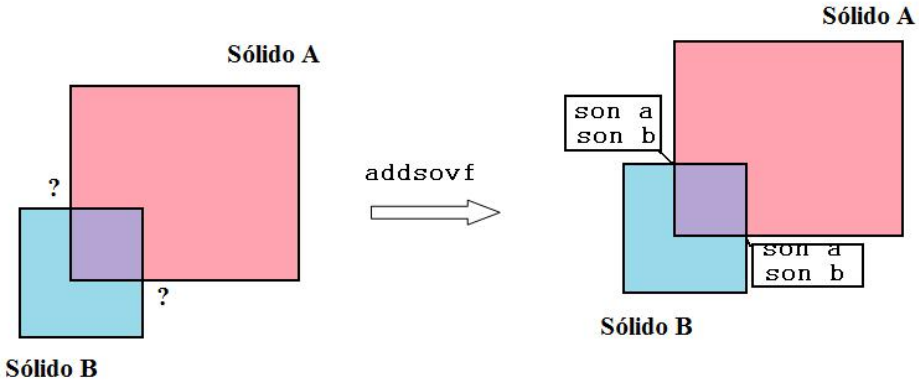


Figura 26 Función `addsovf`

`Addsovv`: Al igual que la función anterior, permite identificar las intersecciones de las aristas del sólido A con las aristas del sólido B para ello utiliza la rutina `AddVeV2Vlist` y es a través de ella que hace la identificación de los nodos hijos de los sólidos y reescribe la lista de los vértices; la variable ahora a delimitar el proceso es `MAXVONV` que es usada en la función `CheckBounds` igualmente para comparar las cantidades.

`bMueveCaras`: realiza un barrido sobre las caras para crear las nuevas caras sobre los nodos hijos, esto lo hace con la ayuda de la función "`movefac`". El barrido es completado con 2 lazos de programación (palabra reservada "`For`") sobre las caras del sólido 1 y del sólido 2 colocando como parámetro de `movefac` las listas de los hijos de ambos.

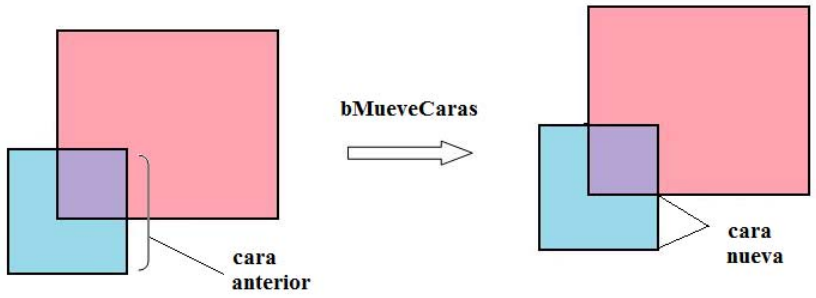


Figura 27 Función `bMueveCaras`

`bSplitNullFaces`: Elimina las caras nulas de las operaciones realizadas, la función parte de los hijos de los sólidos que interactúan y mediante un operador de Euler inverso, específicamente el `OP_EU_lmfkrh` aplicado a los lazos de los hijos de los sólidos, con criterio de parada el máximo número de caras de los sólidos.

`Cutab`: la función permite identificar las intersecciones para su reacomodo según la forma de interacción de ambos sólidos, esto lo hace con los operadores de Euler inversos `OP_EU_lkemr` y `OP_EU_lkef` apoyándose en el máximo número de nodos hijos identificando cada uno y haciendo un barrido a través de las caras, con el término de semiarista (Half edge) quiere decir que recorre en los sentidos para poder cerrar los lazos indicados.

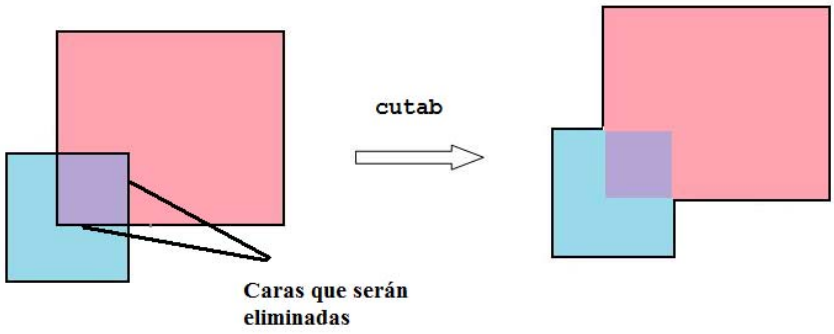


Figura 28 Función `Cutab`

`Dosetopgenerate`: Ubica las intersecciones de las aristas del sólido, para poder identificar si están sobre una cara, sobre una arista o sobre un vértice; de ser así realiza el proceso de modificar el sitio de intersección, mediante el operador inverso `OP_EU_lmev` y las funciones `addsovf` y `addsovv`.

Plantea 3 casos posibles:

Caso 1: La intersección está contenida en una cara, aplica el operador de Euler para la arista transformando la semiarista en una arista temporal y asignado una semiarista temporal, entra en juego la función `addsovf` y redefine el extremo de la semiarista como un hijo del sólido y por último se hace uso de la función `process_edge` que me incluye el hijo del sólido como parte del recorrido de la nueva cara formada.

Caso 2: La intersección está contenida en una arista, de ser así, realiza un doble proceso de operador inverso, para las dos aristas implicadas, nuevamente entra en juego la función `addsovv`, con ello se redefine el tamaño para ambas aristas siendo el límite los hijos de los sólidos, por último la subrutina de `process_edge` que define la nueva semiarista de los sólidos

Caso 3: Se ubica la intersección sobre un vértice, procesa un caso muy similar al primero sólo difiere en los parámetros que ingresan a la función de `addsovv` ya que se toma en cuenta un término denominado "`sonvv`" que es definido como un vértice hijo de otro vértice.

Por último los nuevos vértices creados de estos 3 casos se consolidan a los sólidos a través de la función `DoVertexOnFace`.

`DoVertexOnFace`: Unifica los vértices creados con el nuevo sólido en formación, esto con las funciones `addsovf` pero ahora como parámetros de entrada los hijos de los sólidos y la operación que se está realizando, la función plantea al igual que la anterior 3 casos a tratar cuando en presencia de la intersección del vértice contenido en una cara, del vértice contenido en una arista o en el caso en que el vértice coincida con otro vértice del sólido operando. Las funciones `adsovv`, `addsovf` y el operador inverso de Euler `OP_EU_lmev`.

`InOppSides`: la función es la de ubicar según las semiarista el recorrido de los 2 sólidos o secciones que interactúan esto mediante la comparación de los recorridos de las secciones como lo son positivos o negativos en función de la operación que se haya seleccionado, se hace únicamente con comparaciones entre arreglos de vectores y funciones de vectores como lo es multiplicación de un arreglo por un escalar y la adición o sustracción de los campos.

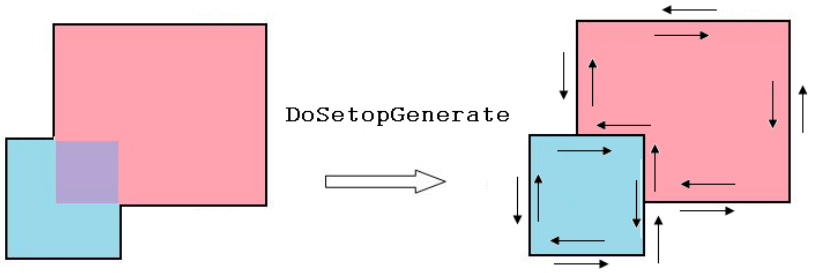


Fig 29 Caso 1 `DoSetopGenerate, DoVertexOnFace y InOppSides`

`InsertNullEdges`: para poder crear las aristas en función de las intersecciones se implementa esta función, ella identifica los nodos hijos y los enlaza con el sólido que está en formación, esto con la función recursiva de `s_insertnulledges` utilizando como parámetros: los hijos de A, los hijos de B, los números de vértices y los arreglos de los mismos (ubicación).

Esto se hace para cada sólido con parámetros de entradas los hijos y la semiarista asociada al nodo, para obtener entonces como salida un arreglo o una cadena de vectores con la información de las semiarista creadas

`Join`: indica paso a paso las intersecciones haciendo uso de los operadores inversos de Euler: `OP_EU_lmef` y `OP_EU_lmekr`, parando en cada punto de intersección, separando los dos lazos de los sólidos. Tiene como criterio de parada el máximo número de caras y crea la nueva cara como un elemento temporal llamado `newf`.

`Movefac`: La función redefine la cara que es afectada por la operación booleana y la reescribe con los cambios que ha sufrido por la intersección, esto lo hace a través de las funciones `DelListFace` y `AddListFace`; tomando como lazo de recorrido las semiaristas del mismo e implementado la rutina "`OtraMitad`" sobre las caras del sólido. Descarta las semiaristas que no interactúan y sustituye la lista de caras con los cambios.

`OtraMitad`: Es una macro que se encuentra definida en un archivo de cabecera, específicamente en el `gwb_macros.h` y se encarga de tomar en el caso de las semiaristas los dos sentidos de recorrido de las mismas por lo que define completamente una arista, quiere decir que al considerar un sentido de recorrido con esta función se estaría considerando el sentido contrario del recorrido.

Esto lo logra definiendo como la semiaristas en función de semiarista 1 y de semiarista 2, siendo 1 y 2 los sentidos de recorrido de las semiaristas, luego al definir el proceso como semiarista 1 y luego "`OtraMitad` (semiarista1)", se estará garantizando el recorrer la arista completa (en ambos sentidos)

`Process_edge`: permite que se guarde la operación Booleana que se ha seleccionado, haciendo la sustitución de las caras temporales parte de los sólidos nuevos por lo que define el resultado de la operación, esto lo hace recurriendo a la función `dosetopgenerate` pero con los datos de entradas de caras y aristas y el sólido BvsA, haciendo un recorrido por cada una de las caras del sólido con un `For`.

`ScanJoin`: propone ir barriendo los vértices de ambos sólidos con el propósito de determinar las semiaristas de terminación del sólido resultante, en pocas palabras la identificación del sólido formado a raíz de la intersección de los anteriores, esto lo hace con la ayuda del número de vértices y de la función `CheckBounds`

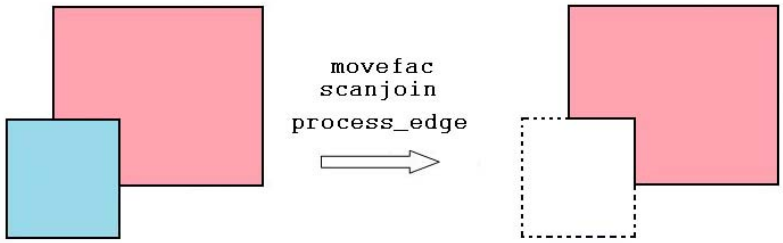


Figura 30 Funciones `Movefac, Process_edge y scanjoin`

`SetopGenerate`:

Este subproceso, mediante los Operadores de Euler [CHEV08] permite crear o dar forma a los objetos que van a interactuar y definiendo los vértices y aristas de cada uno de ellos.

`SetopClassify`:

Esta dividido en dos funciones que permiten jerarquizar las operaciones entre vértices – vértices y vértices - aristas, obteniendo la clasificación de los nodos (Nodo Padre y Nodo Hijo), esto entonces será el paso fundamental a la hora de procesar cualquiera de las 3 operaciones booleanas.

`SetopConnect`:

Este proceso es quizás el más importante ya que va a generar las aristas de intersección que ocurren entre los 2 sólidos, ello me permite entonces definir el espacio de interés, ahora bien, es de suma importancia el proceso que ocurre dentro de esta función , ya que ocurren una serie de llamados a los Operadores de Euler y los Operadores Inversos, con esto define los parámetros que interesan para graficar el sólido (vértices, aristas, caras nuevas y viejas) adicionalmente depura la nomenclatura, dejando la geometría con los puntos que son los que generan mi espacio.

`SetopFinish`:

El último proceso que encontramos en nuestro esquema es el más importante, permite crear el sólido, esto quiere decir que los vértices que una vez fueron hijos ahora son vértices consolidados como principales de igual manera depura el sólido de vértices intermedios en las aristas, todo esto a través de la Función “`LoopGlue`” (definida por Mäntylä [MANT88] como Gluing) o en español la función de pegado.

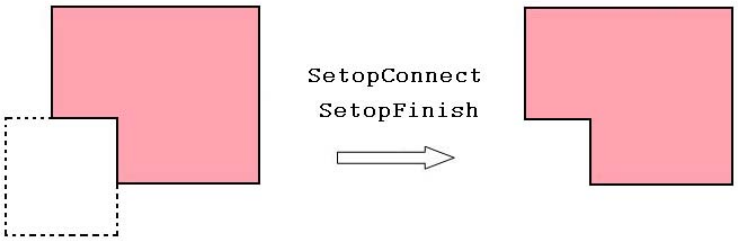


Figura 31 Funciones `SetOpConnect y SetopFinish`

`Solidls` (se encuentra en el archivo `listsolid.c`): esta función es la que permite generar el sólido, a partir del listado de las características de los vértice, caras y aristas, es decir es la función fundamental para decir que una serie de datos son un sólido y caracterizarlo como tal.

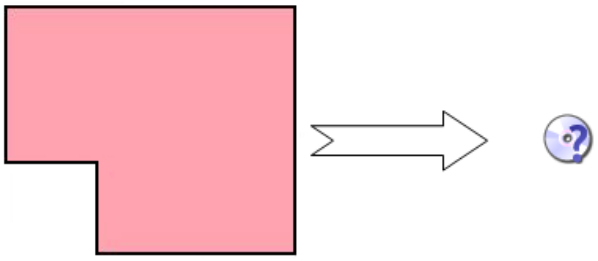


Figura 32 Función `Solidls`

Es importante destacar que los Operadores de Euler son el pilar fundamental del programa ya que la metodología debe crear y/o eliminar aristas en función de los vértices que interactúan, los operadores fueron desarrollados con mayor detalle por [CHEV08] tanto los Operadores como los Operadores Inversos. Si bien no son mencionados dentro de las llamadas del programa, se encuentran en muchos de los procesos antes mencionados.

Para finalizar, el proceso debe almacenar los datos que caracterizan al sólido en un archivo para su posterior manipulación, esto se realiza con un esquema ya definido que incluye la información del número de vértices, el número de aristas y el número de caras del sólido y una lista de ubicación y conformación de cada uno de ellos. La interfaz de la forma en que fue desarrollada permite al usuario definir con que nombre será almacenado el sólido para su llamado.

Dado que las funciones de primitivas desarrolladas en trabajos anteriores están limitadas a la creación de primitivas en posiciones específicas, se implementa un programa adicional que permite modificar los datos de los sólidos, generando como resultado una rotación de 90° del sólido interés, dicho función se puede ubicar dentro del programa como un elemento no vinculado al menú, haciendo el llamado de "voltear".

FUNCIÓN VOLTEAR

Las primitivas desarrolladas por [CHEV08], permiten sólo la creación de los sólidos en una posición en específico (con el cuerpo de los mismos en dirección de Z^{+}), por lo que si el usuario deseara crear un sólido en dirección de otra coordenada, no podría hacerlo. Por ello se ha creado la función voltear, la misma no es una aplicación directa del programa pero es una herramienta útil cuando de estos casos se trata.

La función voltear es compilada a la hora de crear el programa y se puede ubicar en la carpeta Test, su código fuente se ubica en "`voltear.c`" y posee como entrada el nombre de un sólido y genera como resultado el archivo "`volteado.res`".

A continuación se muestra el código fuente de dicha función:

```
#include <stdio.h>
#include <stdlib.h>
#define TAM 200

/*
Función que me permite voltear las columna de datos de los sólidos,
por ejemplo de X a Y o a Z, con solo editar un parámetro abajo
indicado*/
void voltear(double* a, double* b)
{
      double t;
      t = *a;
      *a = *b;
      *b = t;
}

int main(int argc, char** argv)
{
      FILE* f;
      FILE* out;
      char linea[TAM];
      int nvertices, i, ncara, narista, id;
      double x, y, z, temp;
      f = fopen(argv[1], "r");
      out = fopen("volteado.res", "w");
      if(f == NULL)
      {
            printf("No se abrio el archivo\n");
            exit(0);
      }
      fgets(linea, TAM, f);   //ignorar
      fputs(linea, out);
      fgets(linea, TAM, f);   //ignorar
      fputs(linea, out);
      fscanf(f, "%d %d %d\n", &nvertices, &ncara, &narista);
      fprintf(out, "%d %d %d\n",nvertices, ncara, narista);
      fgets(linea, TAM, f);   //ignorar
      fputs(linea, out);
      for(i = 0; i < nvertices; i++)
      {
            fscanf(f, "%d %*c %lf %lf %lf %*c\n", &id, &x, &y, &z);
///paràmetro que se debe modificar para realizar la operaciòn de
volteo
            voltear(&x, &z);
///*************************************************************
*****
```

```
            fprintf(out, "%4d < %15lf %15lf %15lf >\n",id,x,y,z);
      }
      while(!feof(f))
      {
            fgets(linea, TAM, f);   //ignorar

            fputs(linea, out);
      }
      fclose(f);
      fclose(out);
      return 0;
}
```

CAPÍTULO VI

RESULTADOS OBTENIDOS

Ya desarrollado el programa, se plantean a continuación las inquietudes de los alcances del programa en la generación de figuras compuestas, por ello se proponen distintas figuras que permiten probar las capacidades del software final y con la ayuda del programa graficador desarrollado por el Prof. Barragán.

Los ejemplos se colocarán de forma ordenada respecto a la generación (Experimentos) y se mostrarán distintas vistas de un mismo sólido para proporcionar una mejor perspectiva de la representación.

Cada uno de los casos se muestra como una experiencia y está basada en sólidos de lo cotidiano; la complejidad de los mismos se limita por las capacidades del programa.

Experimento 1:

Como primer caso se presenta una operación interesante que ocurre entre un sólido y una esfera, dicha operación es una forma de poder observar de forma rápida el resultado de lo que ocurre al respecto de las operaciones de los Operadores Booleanos.

Se aprecia como primera imagen la unión de la esfera con el cilindro; se tomó una esfera centrada en 0,0,0 y se hace unir a un cilindro centrado en -5,0,0 luego se selecciona el módulo de operaciones booleanas, habiendo indicado una operación de unión entre A = cilindro y B = esfera

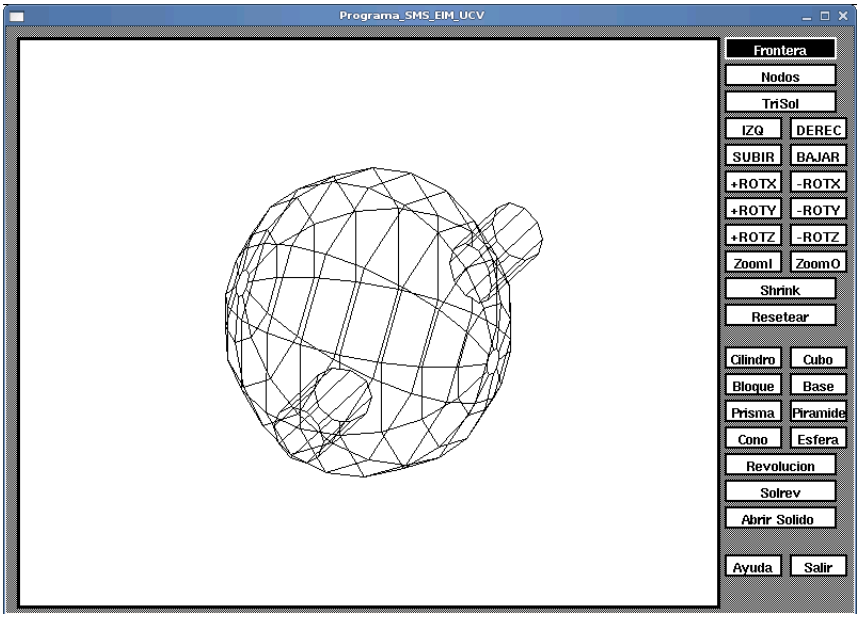


Figura 33 Esfera unida con cilindro

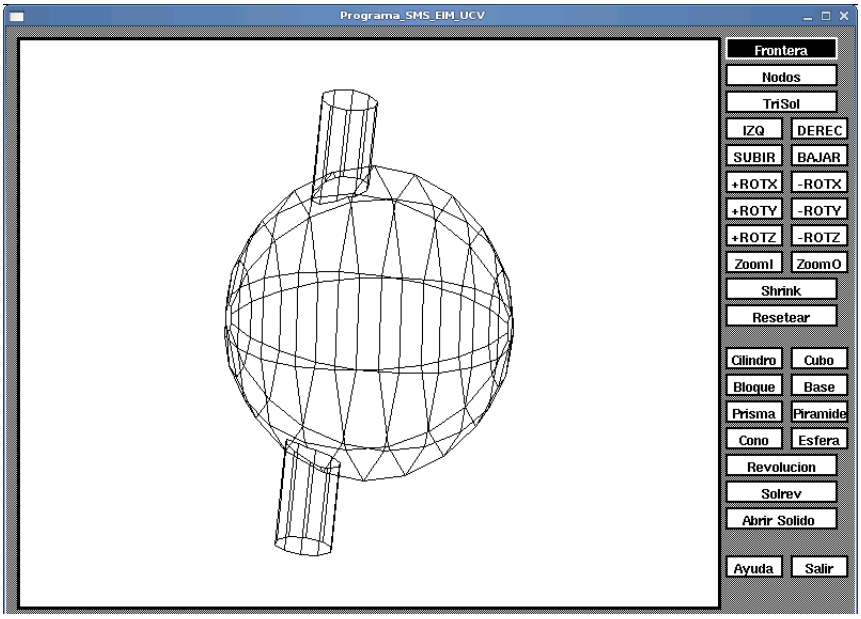


Figura 34 Esfera unida con cilindro 2

Si se toma como segundo caso, la intersección de los sólidos antes mostrados, se presenta a continuación el resultado de la operación:

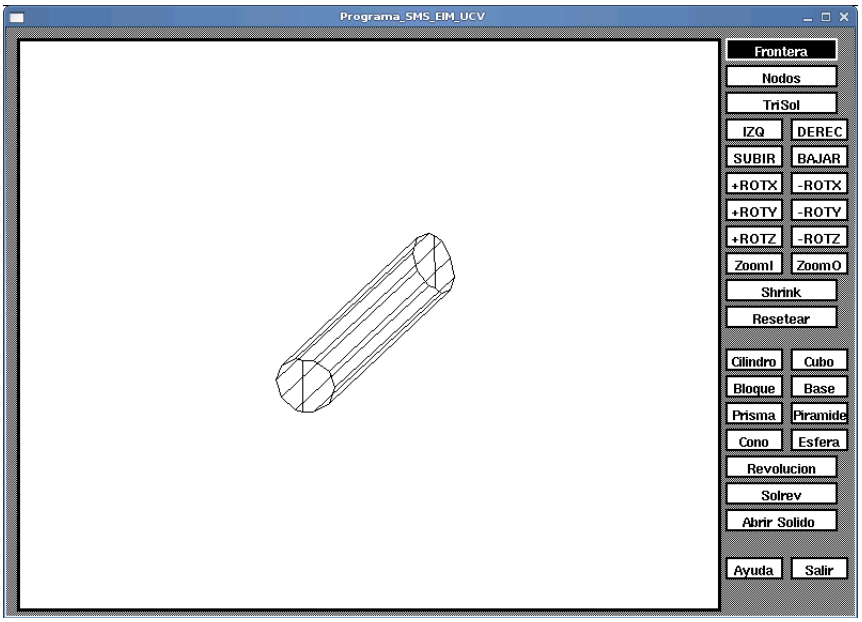


Figura 35 Cilindro intersecado con esfera

Nótese con detalle que ya los extremos rectos del cilindro han sido sustituidos por los extremos correspondientes a la sección de esfera que corresponde a la intersección, planteando entonces un sólido que no puede ser representado por extrusión ni por revolución (funciones que también ofrece el programa).

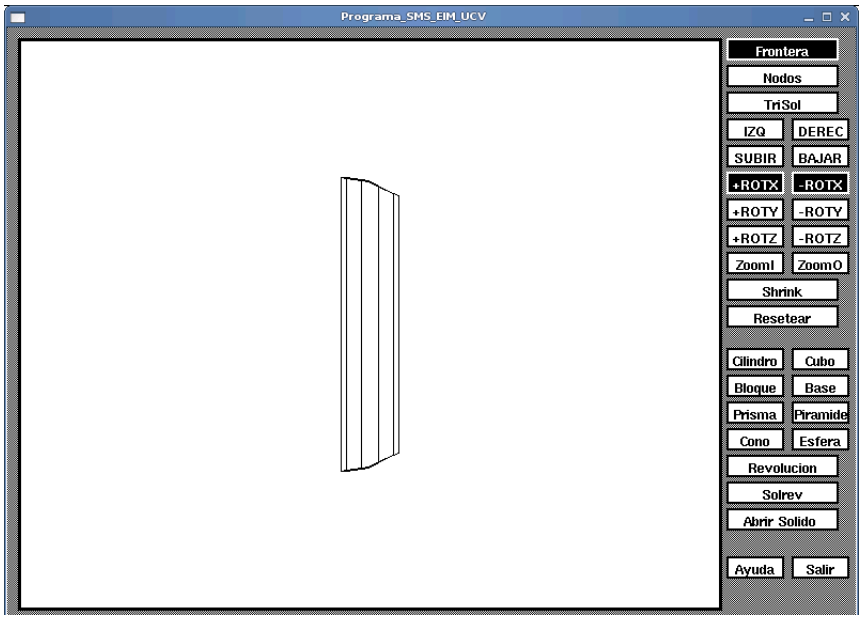


Figura 36 Cilindro intersecado con esfera

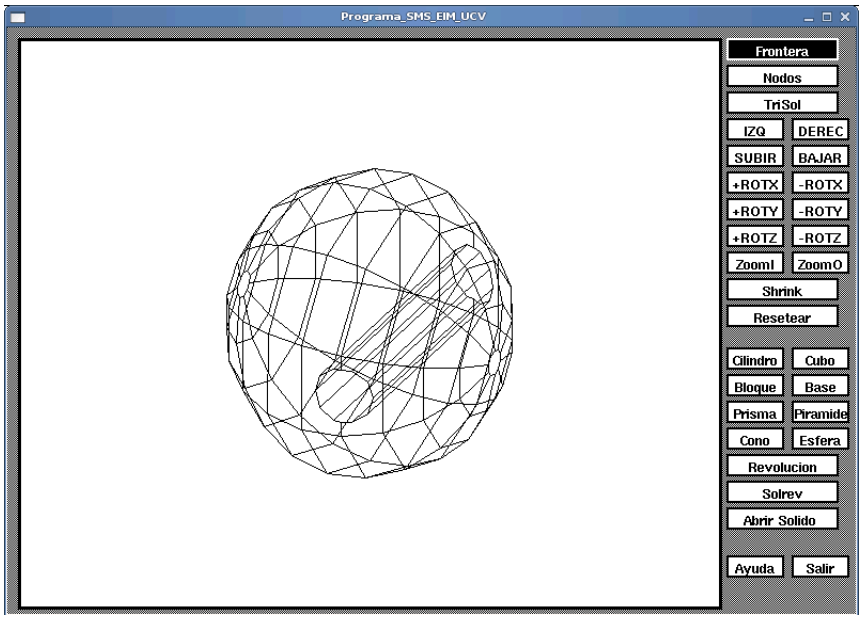


Figura 37 Esfera diferencia cilindro

Al considerar la última operación booleana posible como lo es la sustracción, se muestra en la siguiente imagen el resultado del operador. La esfera en este caso es puesta como sólido base y el cilindro es considerado el sólido Operando, la región del espacio que ocupaba el cilindro desaparece y realmente se tiene una perforación en el sólido base (sólido esfera)

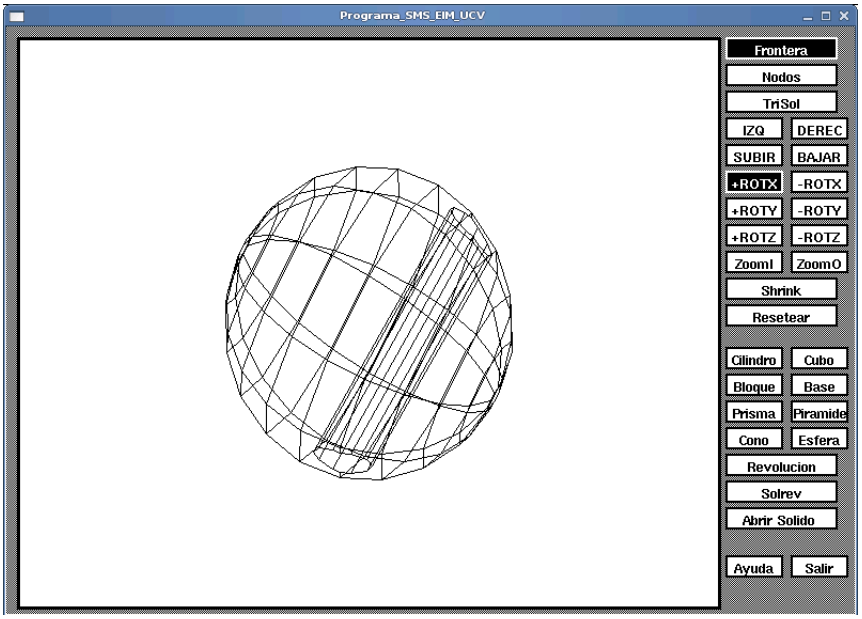


Figura 38 Esfera diferencia cilindro 2

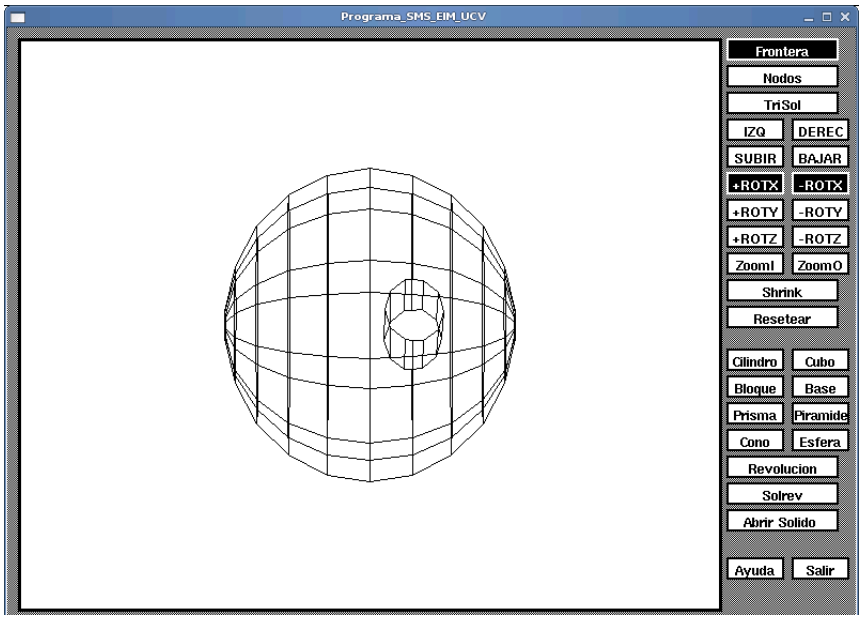


Figura 39 Esfera diferencia cilindro 3

Considerando como sólido base el cilindro y como sólido operador la esfera el resultado obtenido es diferente, se puede apreciar cómo es seccionado el cilindro por la superficie de la esfera, generando entonces 2 nuevos sólidos. Estos sólidos son manejados como uno sólo ya que la información proporcionada de sus vértices y aristas así los relaciona.

Los sólidos resultantes a pesar de ser 2, cumplen con la condición de superficies non manifold y por lo tanto son objetos que cumplirán con las condiciones indicadas al principio del presente trabajo

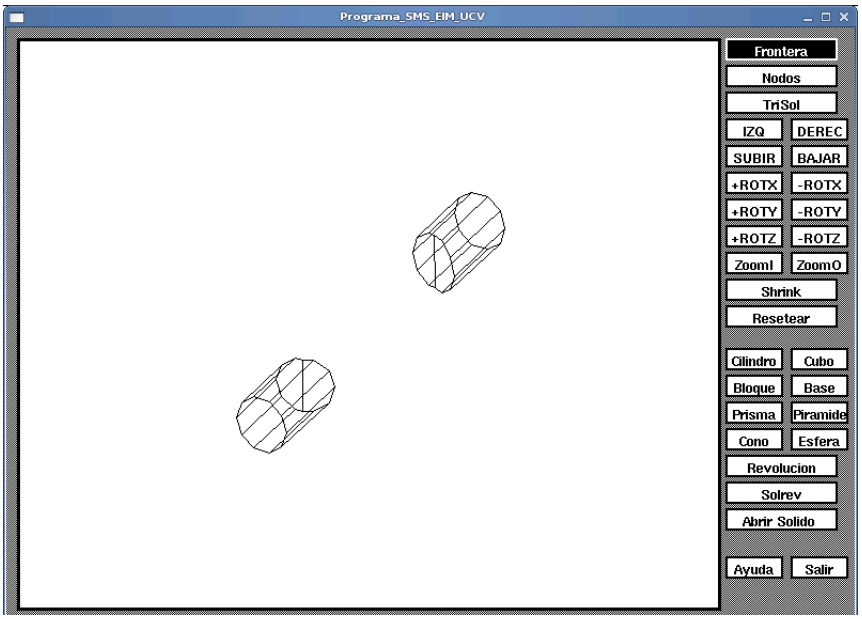


Figura 40 Cilindro diferencia esfera

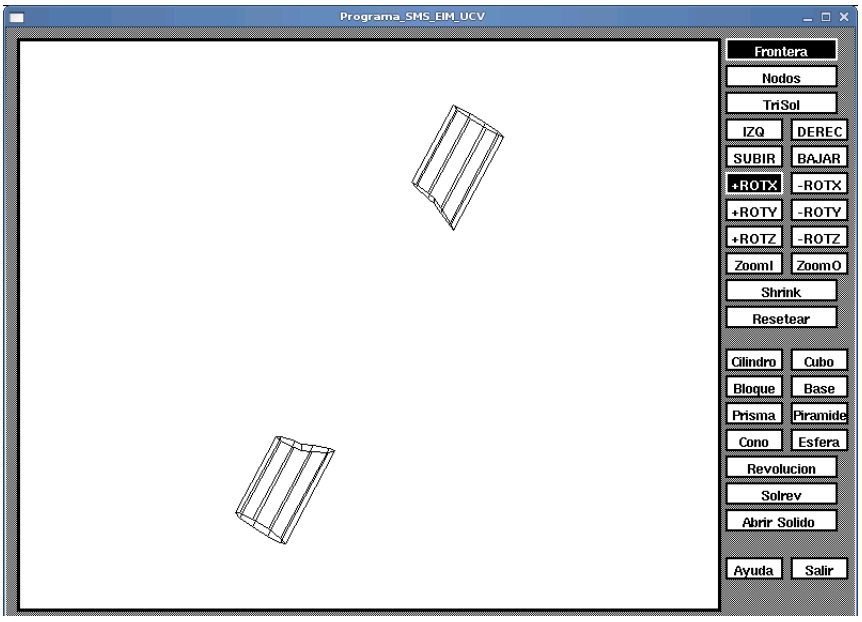


Figura 41 Cilindro diferencia esfera

Experimento 2:

Como segunda propuesta, se tiene la operación de una esfera y un cono, los resultados mostrados a continuación corresponden a la unión de ambos sólidos, estos representan lo que sería una “pelota de bádminton” aunque fue ideada en un principio como una copa o soporte de bola de beisbol.

El mismo es de uso decorativo en aquellos fanáticos que poseen elementos de este tipo y lo desean exhibir entre sus pertenencias

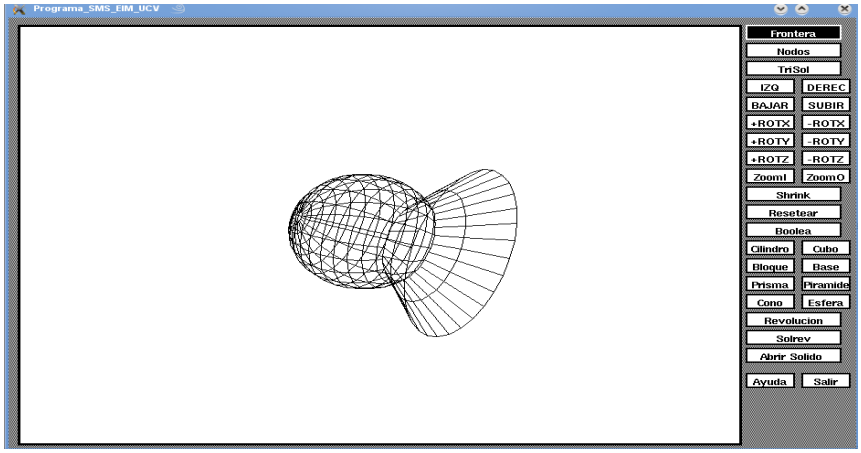


Figura 42 Esfera y cono unión

La forma de crear dicho objeto se obtuvo descentrando los mismos, la esfera se encuentra centrada en la coordenada 0,0,0 y el cono se debió centrar de tal manera que el vértice quedara en 0,0,0 luego se obtuvo que el cono debía centrarse en 0,h/2,0 para poder obtener el resultado mostrado

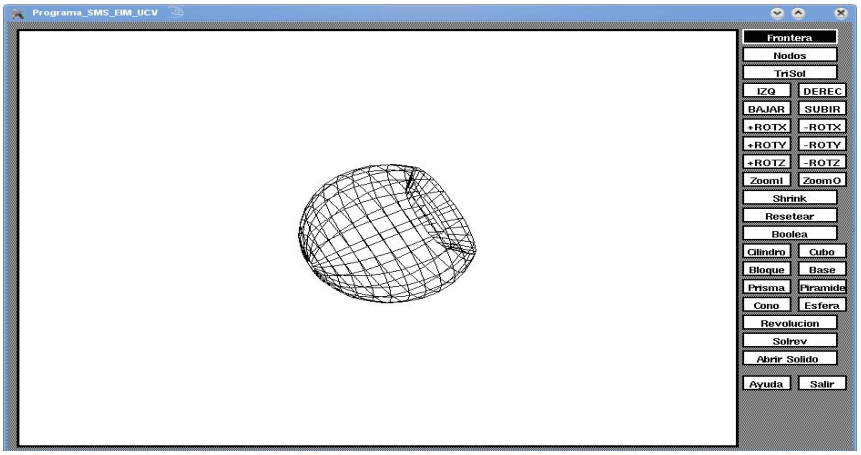


Figura 43 Esfera y cono diferencia

En la figura 45 se aprecia que la diferencia de la esfera con el cono no es completa, quedando como sección de interés la mostrada, inicialmente el propósito era mostrar una esfera con una sección circular faltante (similar a la mostrada en la conocida película “Guerra de las galaxias” como la estación espacial llamada Estrella de la muerte); luego de implementada la imagen, la primera impresión por parte de personas consultadas fue la de una Aceituna Deshuesada o incluso el pomo de una puerta.

El sólido “esfera” carece de los puntos cercanos al eje, la función arco no permite la ubicación de los mismos sobre el eje que se va a rotar por ser cero. De forma computacional no permite ubicar vértices o aristas sobre el eje de rotación.

Experimento 3:

Como 3ra experiencia se muestra un problema que fue planteado hace algún tiempo por el Profesor Rodolfo Berrios (R.Berrios, entrevista personal, Junio 2006), él comentaba sobre un problema existente en la mecánica de fluidos acerca de modelar un espacio como un cubo con una esfera interna como operador booleano.

El propósito de esta tercera experiencia era el resolver dicho problemas por ende se modela el cubo con la esfera interna pero ésta última debe tener intersección con el cubo que la delimita sino de lo contrario el programa detectaría la no intersección y por ello no habría regla de correspondencia entre los dos sólidos que se quieren mostrar u operar. A continuación se muestra el resultado de dicha experiencia:

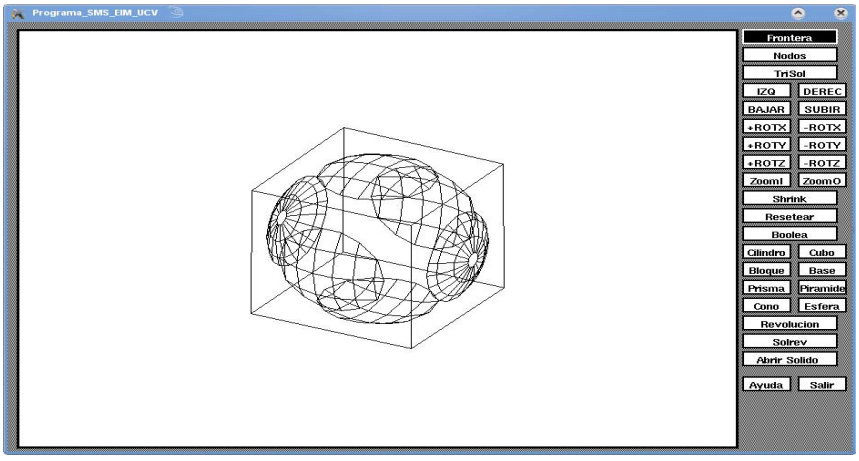


Figura 44 Cubo y esfera unión.

Se presenta una vista en donde se puede apreciar mejor las dimensiones del cubo en relación de la esfera, para que exista al menos una intersección entre los dos sólidos.

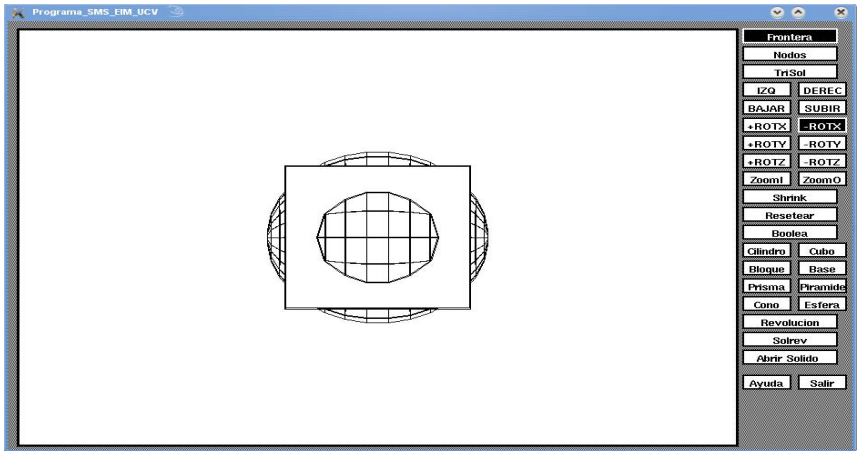


Figura 45 cubo y esfera unión 2

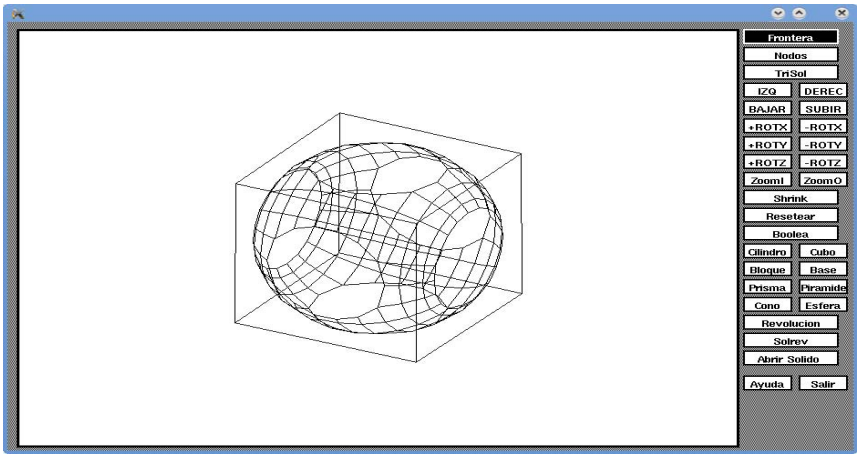


Figura 46 Cubo y esfera diferencia

La figura 48 y 49 muestran en perspectiva el resultado de restarle la esfera al cubo, con el objeto de resolver la inquietud planteada por el Prof. Berrios, mas sin embargo se imposibilita por los momentos dicha operación.

El sólido obtenido corta los extremos de la esfera dejando un espacio que se aproxima de forma esférica entre las 6 caras del cubo mostrado.

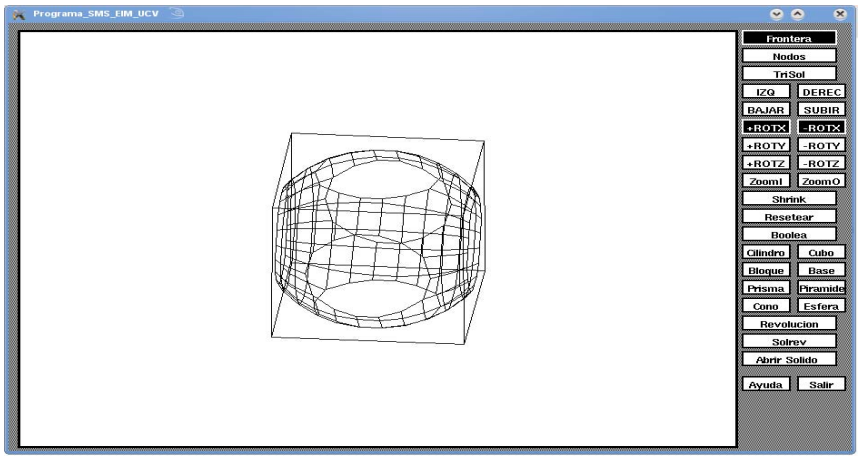


Figura 47 cubo y esfera diferencia 2

Ajustando un poco la precisión de la esfera con que es llevada dentro del cubo se presenta el siguiente ejemplo. Se puede observar como la esfera se compenetra con mayor exactitud que el ejemplo anterior dentro del cubo mostrado.

Si se observa la figura 51, los restos de la esfera tomando como operador la diferencia con el cubo, las secciones de la esfera sobrantes son de menor tamaño que en el ejemplo anterior.

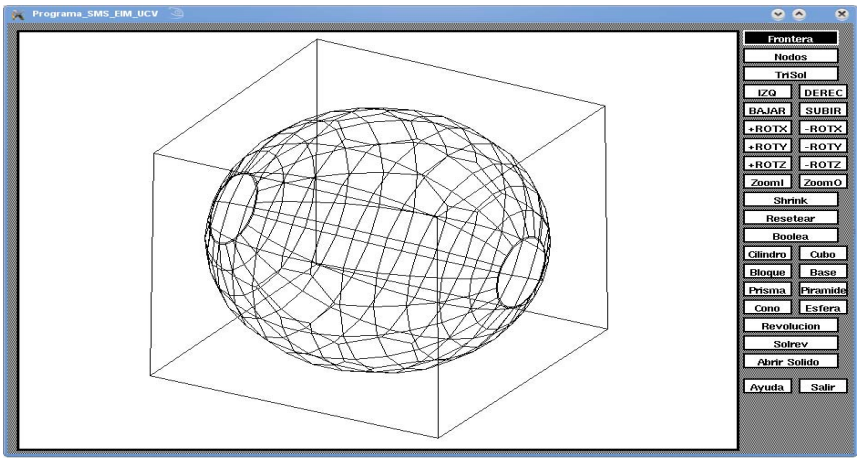


Figura 48 Cubo y Esfera diferencia 3

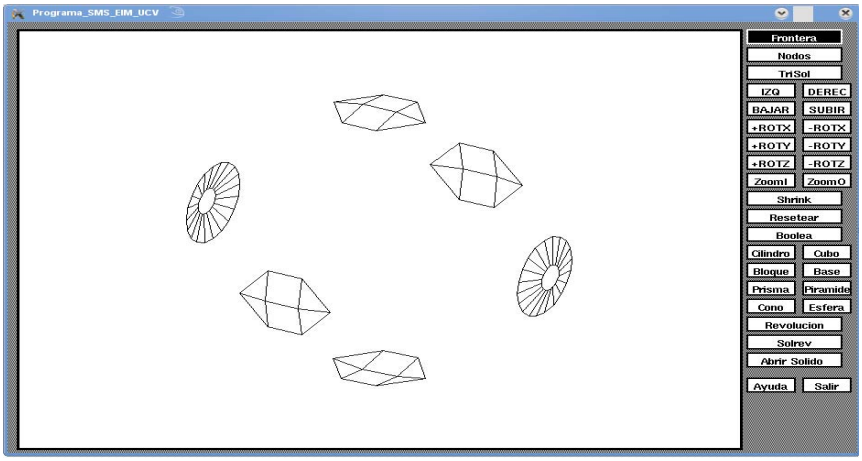


Figura 49 Cubo y esfera diferencia

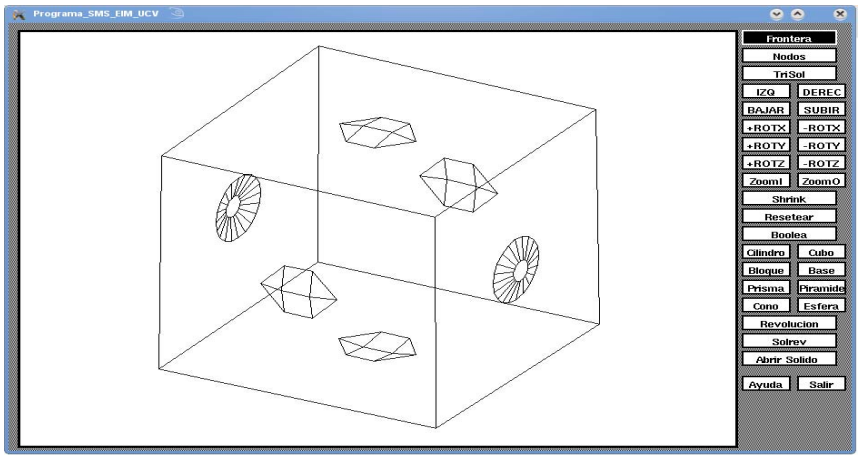


Figura 50 Cubo y esfera unión 3

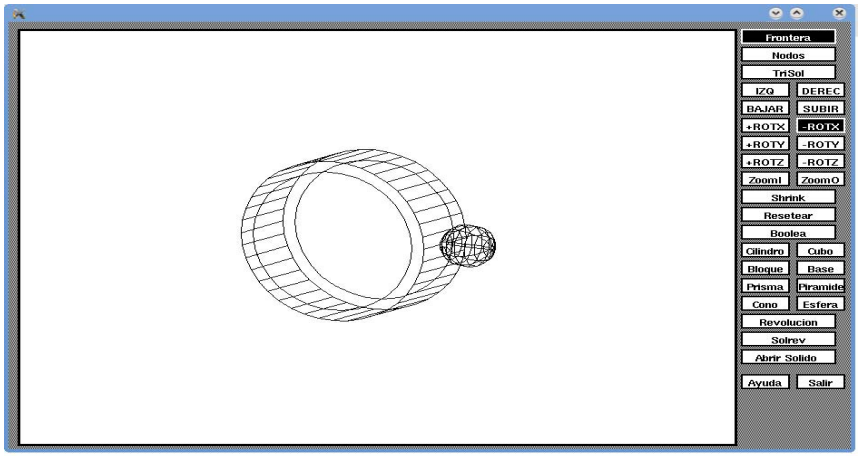


Figura 51 anillo

Experimento 4:

Los objetos de la vida cotidiana también pueden ser trabajados con el programa que se presenta, por ello se decide para mostrar dicha cualidad presentar un ejemplo muy representativo de la versatilidad de la aplicación

La guitarra que se muestra es la consecuencia de 4 operaciones booleanas de forma recurrente: la base, el cilindro (diferencia), el cubo (unión), el mango y la unión de todos los elementos.

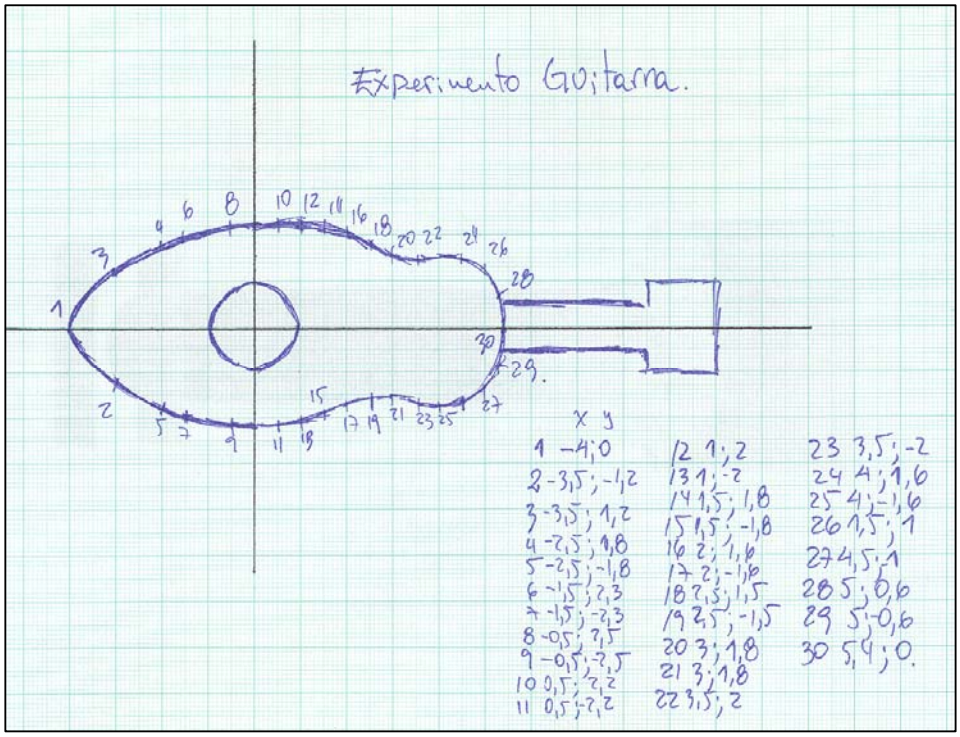


Figura 52 Experimento 4 Guitarra (boceto)

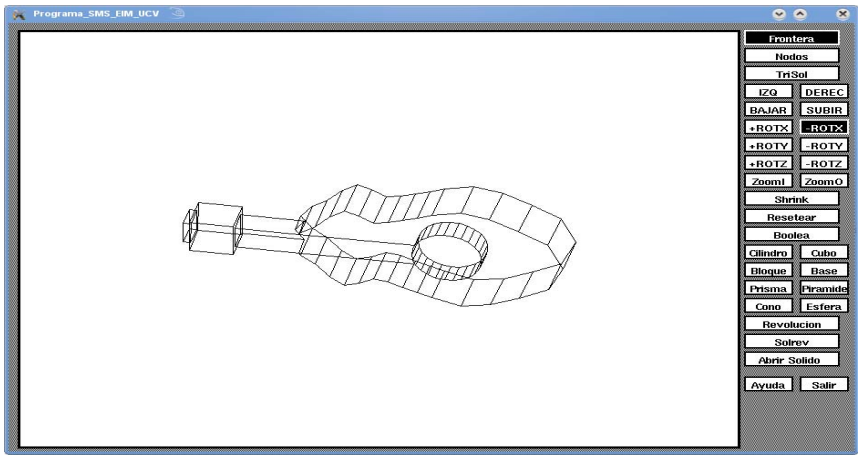


Figura 53 Guitarra

Varias perspectivas del elemento “Guitarra” permiten apreciar el detalle del sólido para poder explotar las cualidades de precisión de la aplicación.

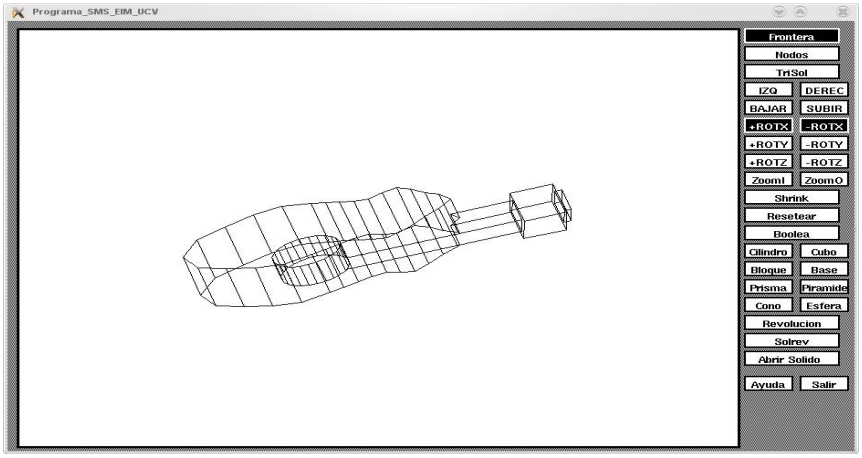


Figura 54 Guitarra 2

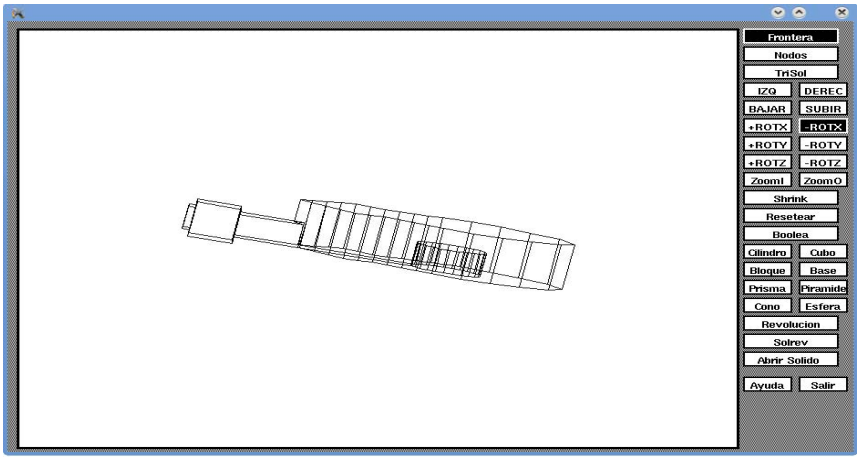


Figura 55 Guitarra 3

Aplicaciones Ingeniería

Experimento 5:

La aplicación de extrusión de la biela mostrada por [CHEV08] podía asemejarse un poco más a la realidad con 2 operaciones booleanas que aquí se muestran: la creación del espacio para la ubicación del cigüeñal y el orificio para la ubicación del pasador del pistón; este ejemplo se desarrolló con la intención de complementar el sólido mostrado anteriormente.

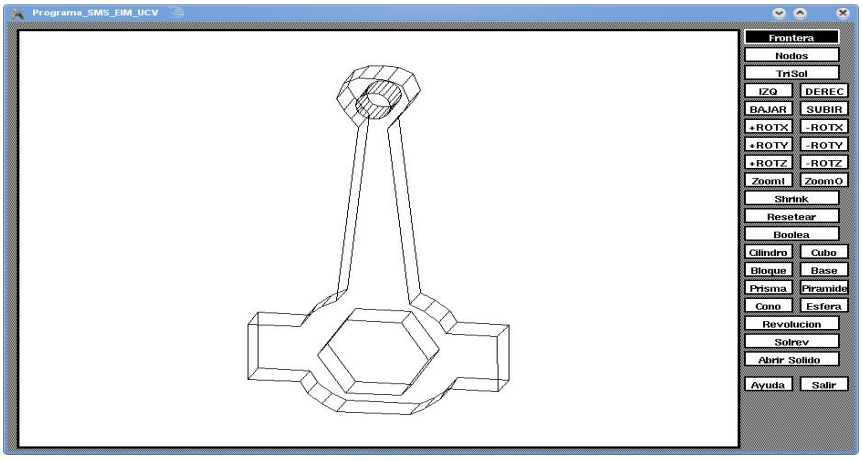


Figura 56 Biela

Experimento 6:

Las herramientas que comúnmente encontramos en cualquier taller mecánico son la llave de trinquete (o matraca) y el dado, que permiten el aseguramiento de pernos de cabeza hexagonal de distintos tamaño.

El experimento 6 consiste en modelar dichas herramientas por su importancia y características para probar las cualidades de la aplicación desarrolla, obteniendo como resultado (cilindro y la diferencia del hexágono y el cubo) las imágenes que se muestran a continuación

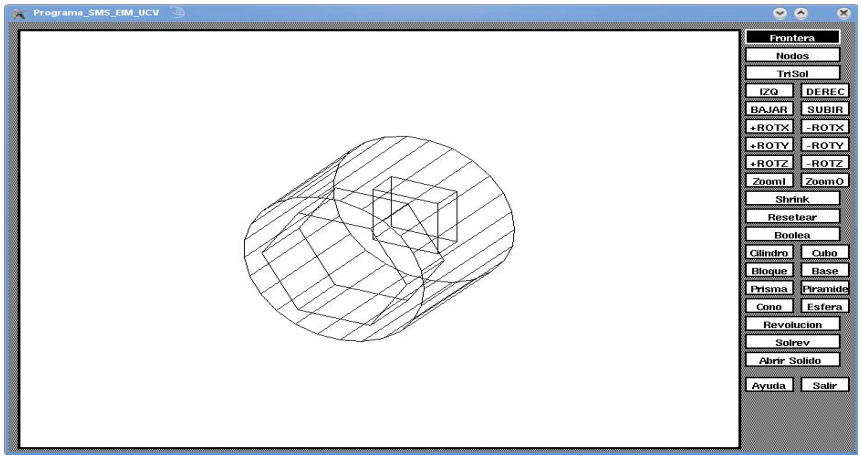


Figura 57 Dado

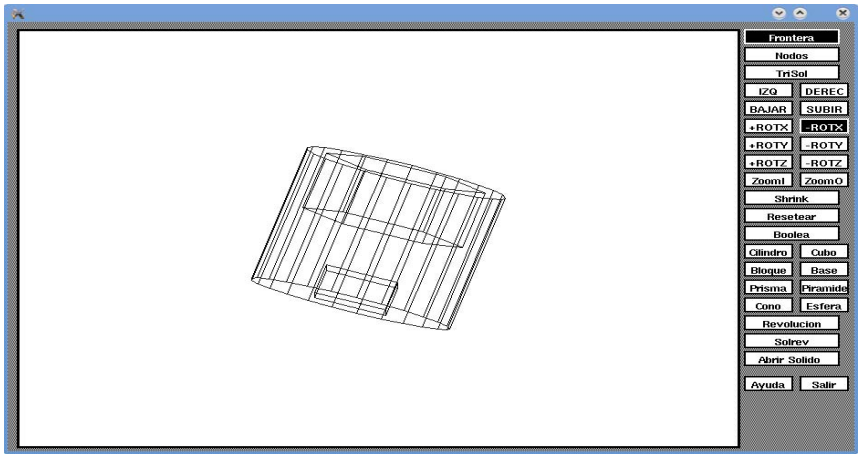


Figura 58 Dado 2

Experimento 7:

Como parte de los sólidos que se busca sirvan de prueba para la aplicación, se presenta a continuación el disco de frenos originalmente del vehículo Chevrolet Cavalier 97.

Figura 59 Disco de Freno (foto)

Nótese la gran similitud entre el objeto real y la simulación realizada con la aplicación presentada, dicha aplicación permite las distintas vistas del sólido y por ello se verifica lo cercano del modelo con la realidad del objeto

Se obtiene el sólido mostrado a continuación, haciendo la unión de los dos cilindros externos y la diferencia de los 6 cilindros internos

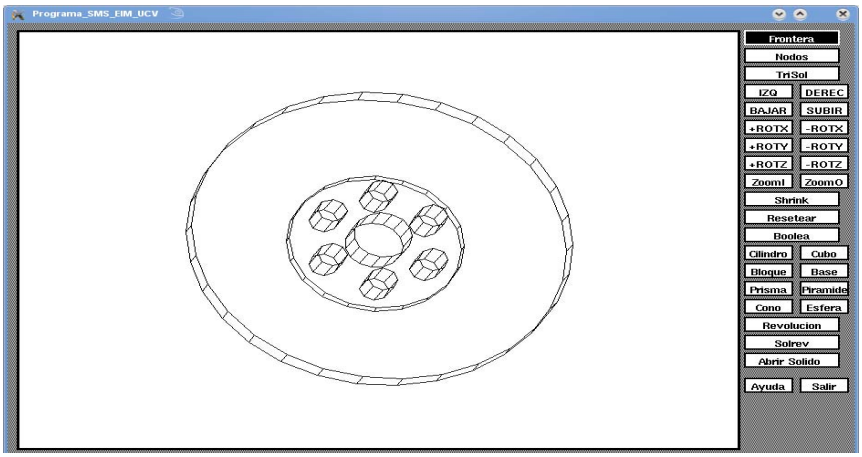


Figura 60 Disco de Freno

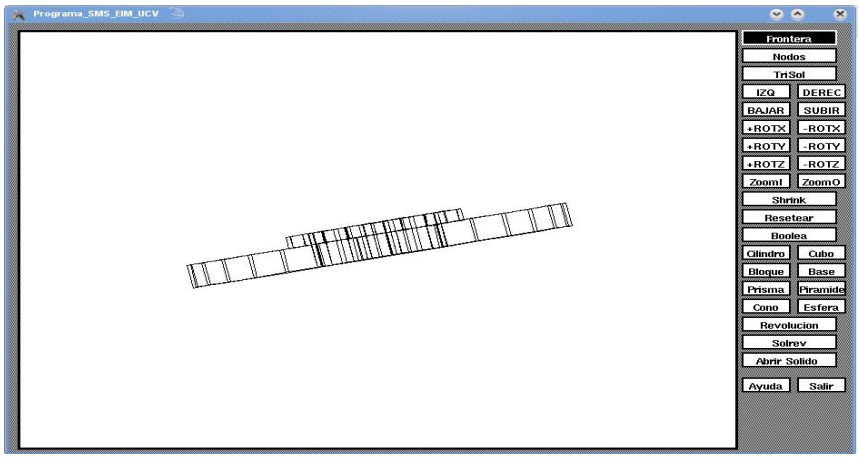


Figura 61 Disco de Freno 2

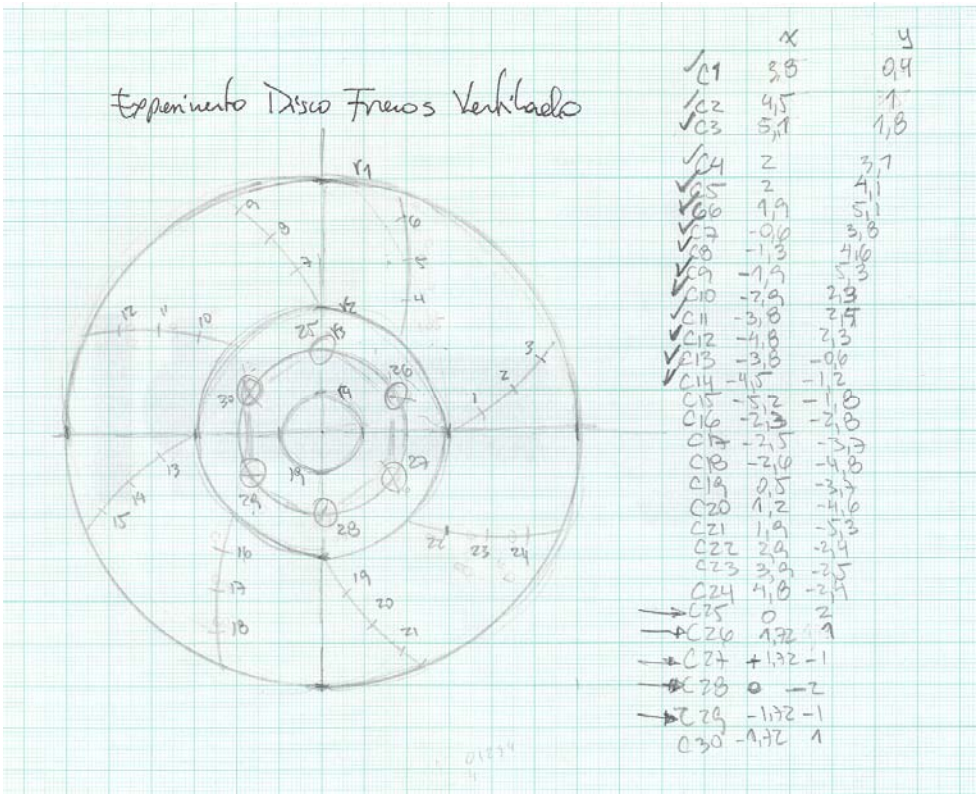


Figura 62 Experimento 7 Disco de Frenos Ventilado (boceto).

Debido a las diversas aplicaciones que poseen los discos de frenos, se intenta modelar el disco de freno ventilado, cuya característica a diferencia del anterior es la de poseer agujeros pasantes en la superficie del disco con el objeto de permitir la disipación de calor; éste ejemplo fue imposible completarlo debido a las limitaciones del programa y aunque se muestra la intención que se tiene, sólo un lado del mismo puede mostrar con el actual programa de gráficos.

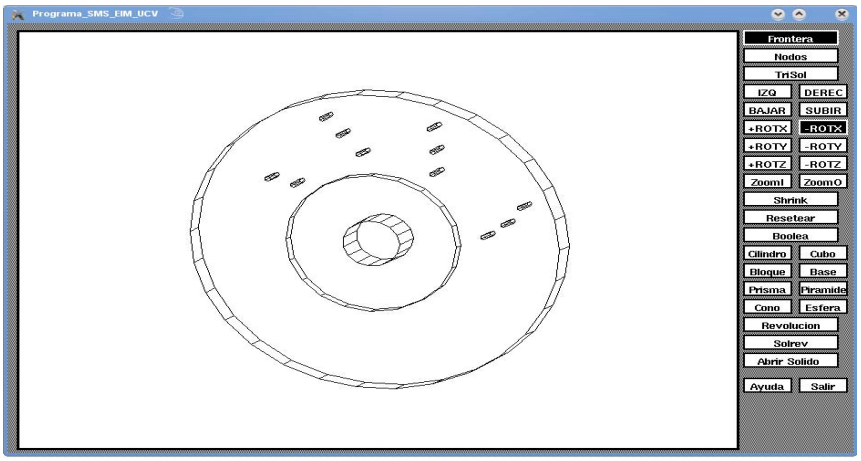


Figura 63 Disco de Freno ventilado

Las medidas del disco aquí mostrado son similares al ejemplo anterior y corresponden al vehículo mencionado. Los agujeros fueron colocados de manera distribuida en la superficie de forma de cubrir cada 60° el objeto cumpliendo así con la forma real de los discos aquí mostrados

Experimento 8:

Una herramienta indispensable en un vehículo es un elemento que permita la sustentación a la hora del reemplazo de un caucho, por ello se elije como experimento el mostrar las cualidades del programa simulando un gato tipo tijera.

Se implementan 3 operaciones booleanas mostrando una superficie extruida como el cuerpo del elemento, un cilindro como el tornillo sin fin (tornillo de potencia) y un tercer elemento cilíndrico a modo de pasador que permite la rotación o el giro del tornillo de potencia.

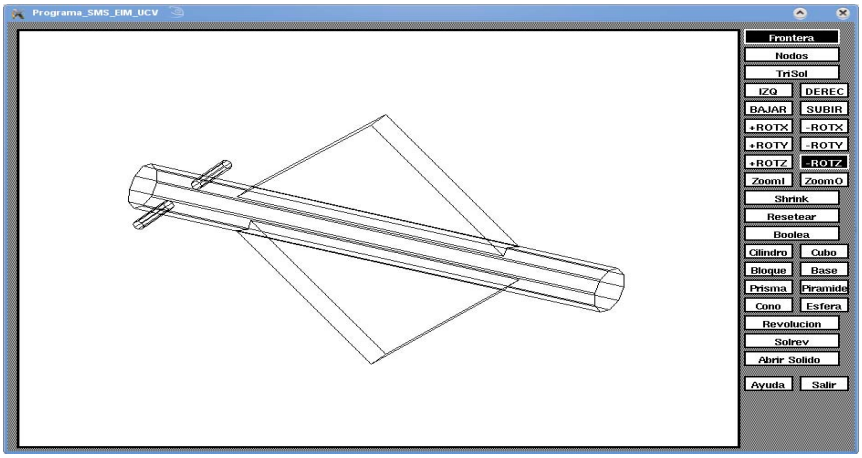


Figura 64 Gato tipo tijera

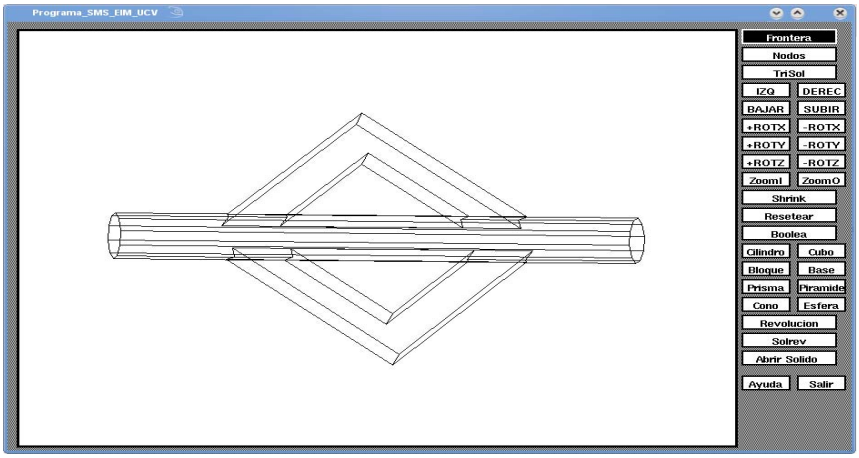


Figura 65 Gato tipo tijera 2

Experimento 9:

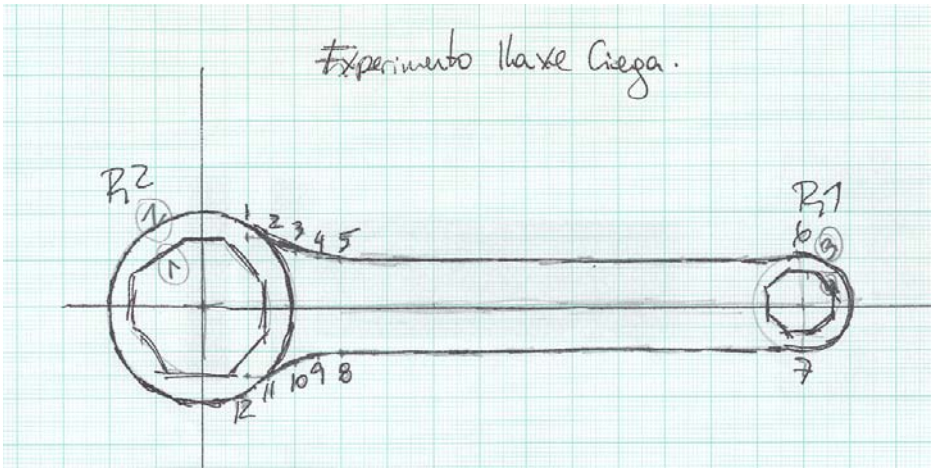


Figura 66 Experimento 8 Llave Ciega (boceto)

Otra de las herramientas importantes es la llave ciega, se obtiene al extruir la superficie de la llave y luego con 2 operaciones booleanas de diferencia, se obtiene los agujeros pasantes generando como resultado las simulaciones mostradas

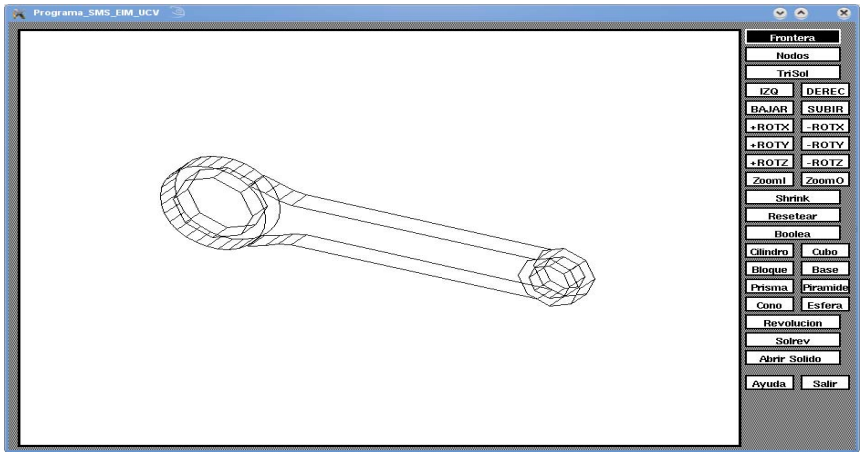


Figura 67 Llave ciega

El modelo mostrado como llave ciega permite de igual forma verificar la precisión del programa, basado en la precisión de los elementos circunferenciales tanto internos como externos

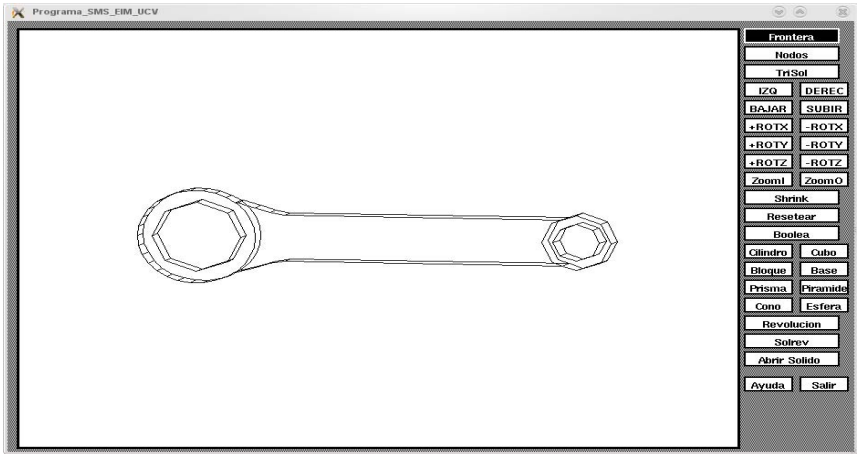


Figura 68 Llave ciega 2.

Experimento 10:

La llave de trinquete o matraca complementa en conjunto con el dado las herramientas indispensables de cualquier cajón de trabajo, la precisión con que es modelado el objeto podemos compararlo con las fotos tomadas

Figura 69 Llave de trinquete o matraca (Foto)

Para la realización de la figura, se extruye la superficie del cuerpo, se hacen 3 operaciones de unión (el mango de la herramienta, el cilindro que forma el cabezal y el cubo)

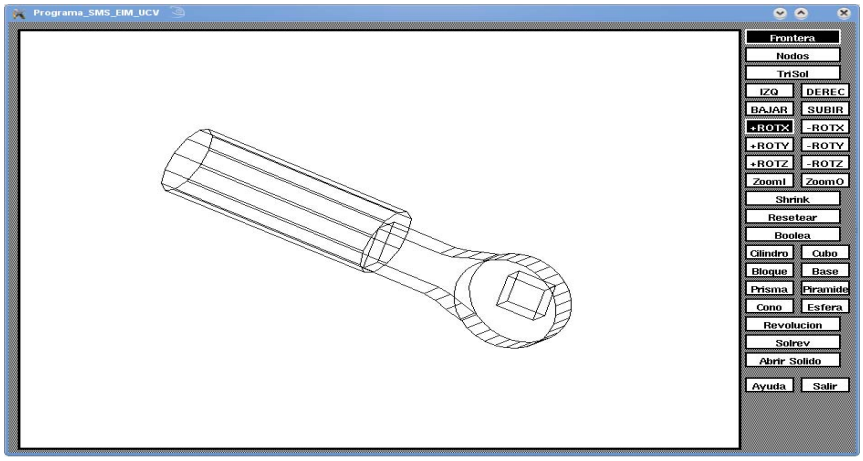


Figura 70 Matraca

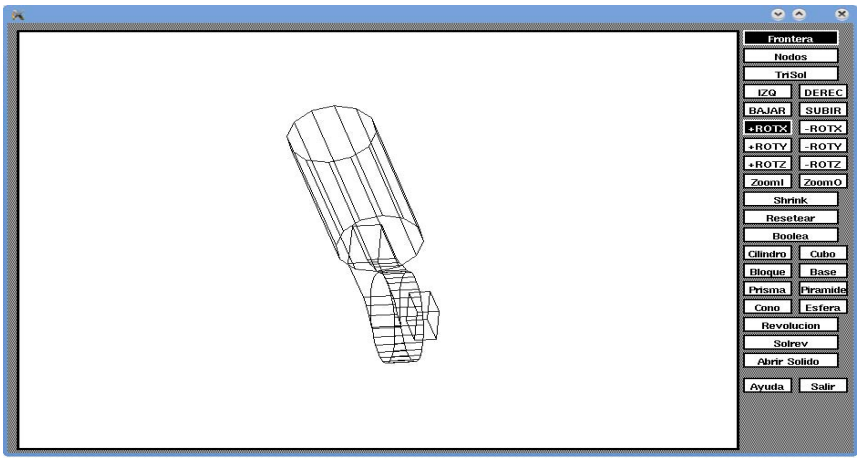


Figura 71 Matraca 2

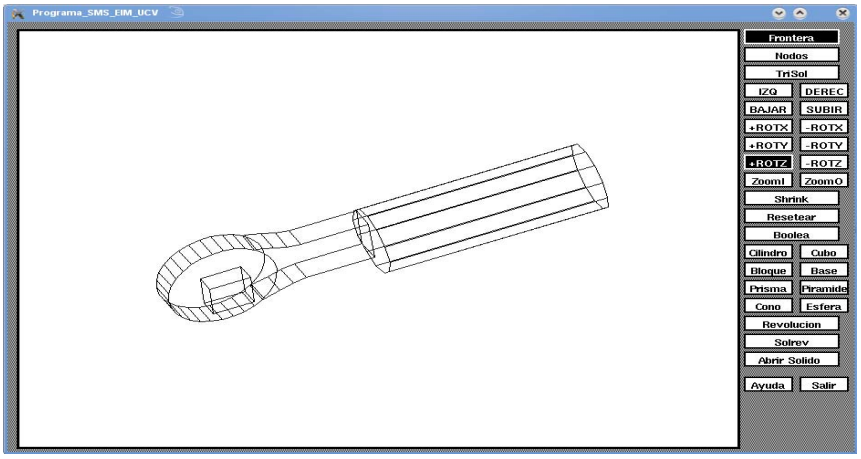


Figura 72 Matraca 3

Experimento 11:

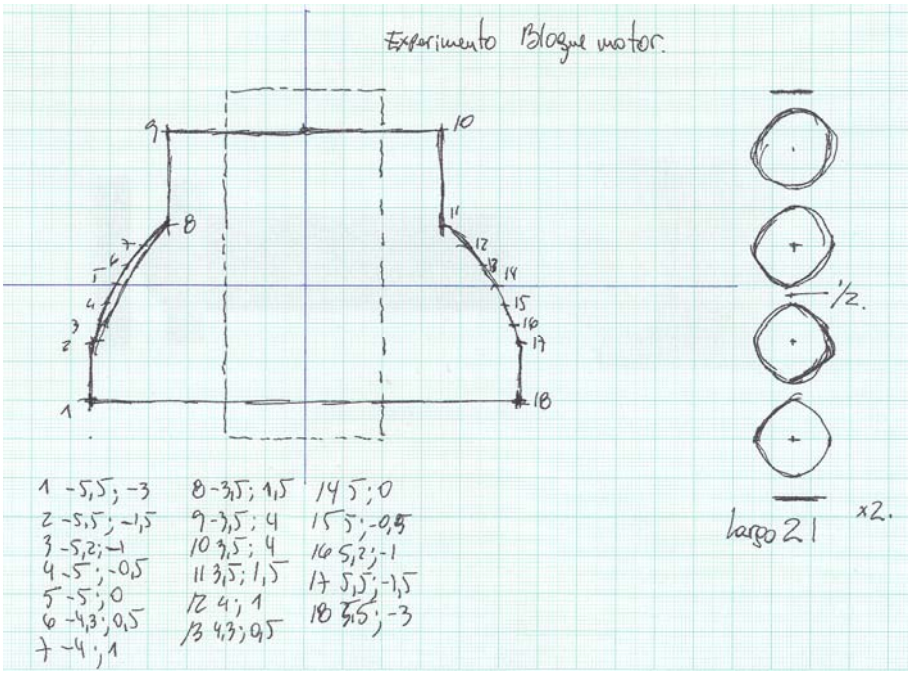


Figura 73 Experimento 11 Bloque de Motor (boceto).

Un ejemplo representativo de las operaciones booleanas se muestra a continuación: el bloque de motor. Consecuencia de una superficie extruída y 4 operaciones booleanas recurrentes, con 4 cilindros centrados en las ubicaciones de los pistones, se obtiene la figura que se muestra a continuación.

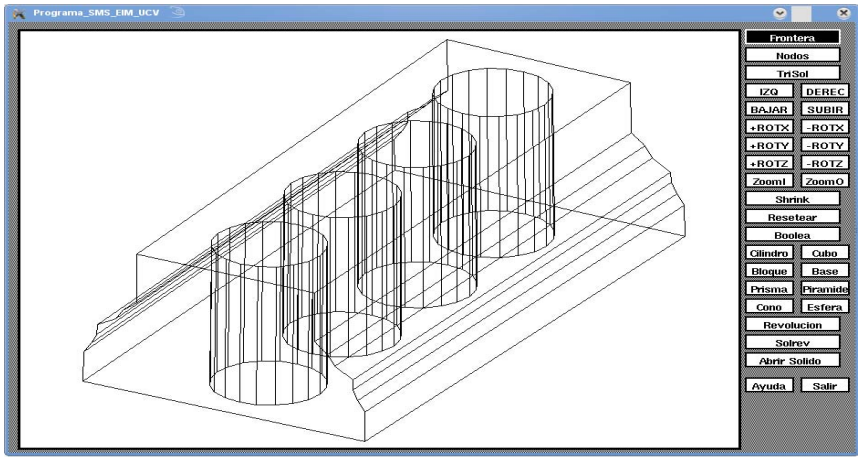


Figura 74 Bloque motor

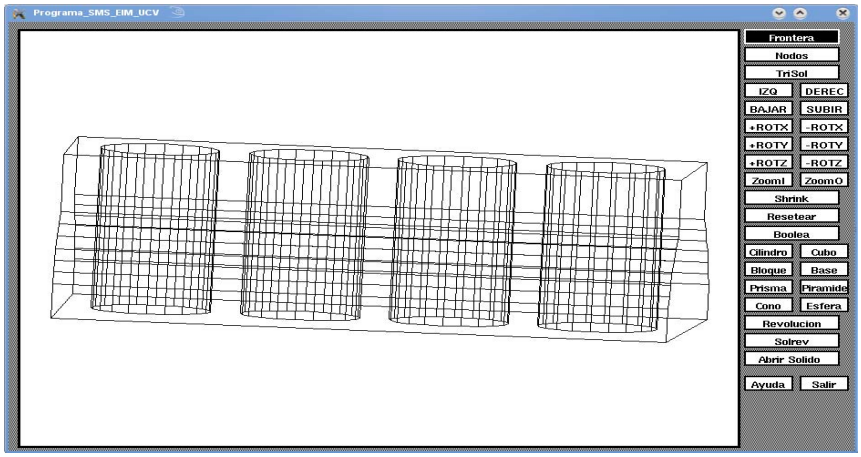


Figura 75 Bloque motor 2

El bloque de motor presentado fue diseñado con medidas hipotéticas y sólo para verificar las cualidades de la aplicación de los operadores booleanos, la verificación se hace de forma satisfactoria al permitir obtener un modelo muy

similar a la realidad, se obtiene de extruir la superficie del bloque y hacer la diferencia de los 4 cilindros que formarían el espacio de los pistones

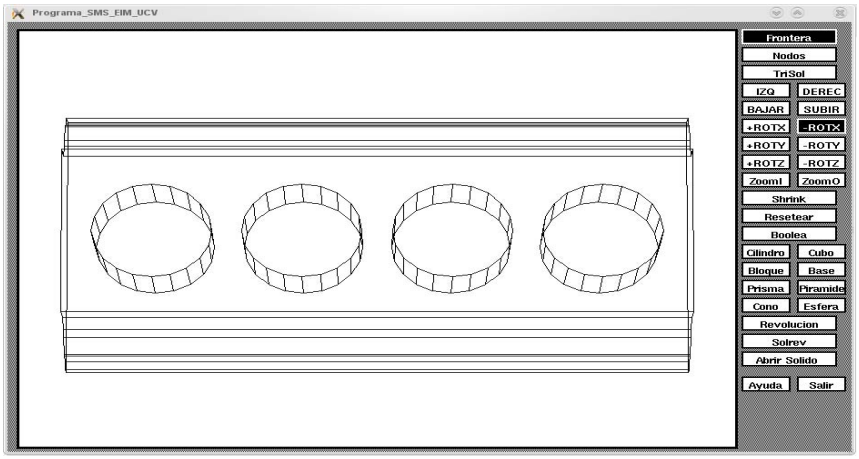


Figura 76 Bloque motor 3

Experimento 12:

El agujero pasante para el pistón desarrollado por CHEV08 permite un mejor acercamiento a la realidad del modelo, a continuación se presenta la foto de un pistón:

Figura 77 Pistón (foto)

La similitud de la simulación permite entonces la manipulación casi completa del modelo hasta la precisión del objeto real, el propósito del modelador de sólido es satisfactorio para el caso del experimento 12.

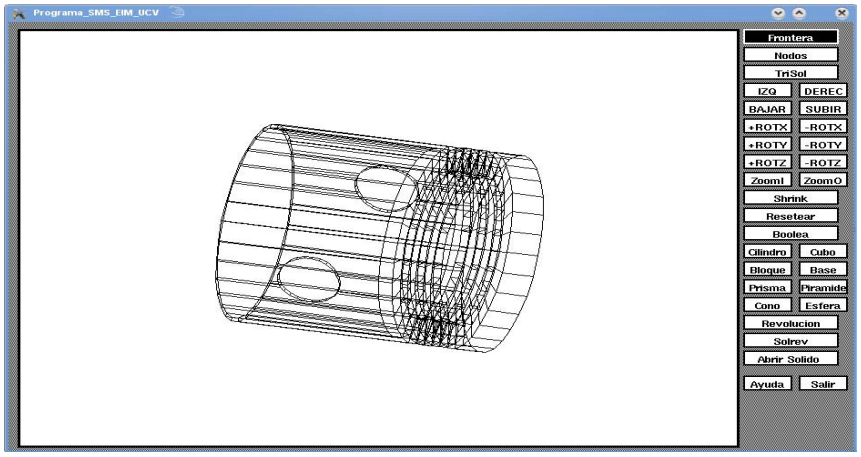


Figura 78 Pistón

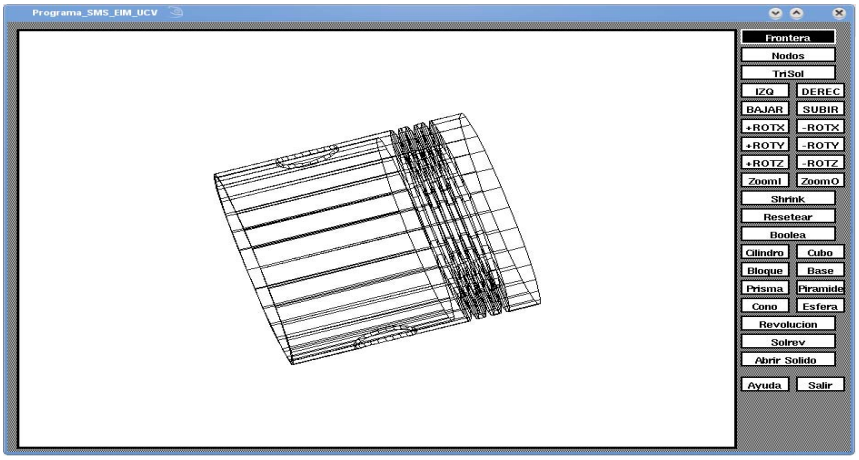


Figura 79 Pistón 2

CONCLUSIONES

El Módulo de Operación Booleana permite al usuario realizar operaciones de sólidos como lo son: Unión, Intersección y Diferencia. Para la generación de sólidos más complejos; así mismo dispone de la posibilidad de guardar y llamar los archivos ya utilizados para hace recursivas operaciones booleanas en caso de requerirse

El módulo de interfaz gráfica que hace de manera más amigable y sencilla la interacción con el usuario mantiene lineamientos de trabajos anteriores con la creación de un botón para desplegar el módulo de operaciones booleanas, que se presenta en el mismo formato que las funciones mencionadas anteriormente.

Se mantienen los botones de manipulación de sólidos, cuyo propósito es el de acercar o alejar la vista del sólido para hacer énfasis en una zona interés por parte del usuario, el desplazar el sólido en las distintas direcciones y la identificación de los nodos del sólido.

La implementación del Sistema de Modelado de Sólidos desarrollado bajo los postulados de Mäntylä permite verificar que los sólidos obtenidos cumplen con las características deseables de las reglas de creación de sólidos como se explicó en la sección de definiciones: Ambigüedad, Validez, Integridad, entre otros.

El haber desarrollado la aplicación de Modelado de Sólidos bajo Unix, específicamente LINUX (OpenSUSE 11.1) genera como principal beneficio el acceso al código fuente, con ello la posibilidad de a través de conocimientos de programación, la mejora o actualización de las aplicaciones que aquí se muestran. El hecho de ser desarrollado bajo estándares de programación AINSI/C99 y un orden de

las funciones, permitirá a aquellos usuarios que deseen su modificación, realizarla sin ningún tipo de dificultad.

Éste programa es desarrollado como compromiso de un Trabajo Especial de Grado y forma parte de la herramienta que debe componer la formación de un Ingeniero Mecánico, por ello posteriormente la ampliación de dicho programa permitirá la integración de los conocimientos de programación con las futuras aplicaciones que se deseen agregar ya sea como Trabajos Especiales de Grado o como desarrollos personales de los usuarios que deseen aportar algo al desarrollo de la presente aplicación.

Los errores obtenidos durante el desarrollo de la aplicación son errores de las capacidades del programa (violaciones de segmento) en la documentación sobre este error se refiere al desbordamiento de memoria por parte de algún proceso que se esté implementado de manera incorrecta o de una aplicación que lamentablemente sobrepasa los límites de memoria destinados, es por ello que los objetos con mayor información (vértices, aristas y caras) que los mostrados en el presente trabajo, deberán ser representados luego de una revisión tanto de las rutinas del programa como de la aplicación de dibujo desarrollada por el Prof. Barragán.

La precisión con que la aplicación maneja los elementos, quedó evidenciada en los experimentos; incluso al querer modelar aplicaciones como discos de frenos, llaves ciegas y algo tan cotidiano como una guitarra permite opinar acerca de la versatilidad de las aplicaciones que puede desarrollar el programa presentado

El desarrollo de las aplicaciones de Modelado de Sólidos dan un valor adicional a la hora evaluar un proceso de manufactura, esto lo podemos observar en aquellas aplicaciones automatizadas cuyas representaciones son muy similares a la muestra de las figuras que aquí se plantean: la orientación de los recorridos, la

ubicación de los vértices y sobre todo los espacios del sólido (en función de los vectores normales a las caras).

La información o documentación que ayudó a solucionar muchas de las inquietudes que aquí se presentaron se obtuvo gracias a la comunidad unida de personas que utiliza el Linux y sobre todo aquellos que dominan el lenguaje C y tienen el aprecio de ayudar a otras personas a través de foros de programadores o foros públicos en la red, si bien hay temas cuyo objetivo es el de profundizar sobre una rutina, existen caminos que permiten que dicha rutina sea aplicada al problema sin necesitar un conocimiento amplio sobre la programación.

RECOMENDACIONES

Una vez presentado el software, nos damos cuenta que existen muchas cualidades de la que se podría seguir explotando material, a continuación se presenta un breve listado de mejoras que quizás en posteriores Trabajos Especiales de Grado se pudieran desarrollar, todo esto a criterio del autor del presente Trabajo Especial de Grado:

Captura de video: si bien es importante el manejo de la información (entrada de datos) se pudiera considerar una entrada gráfica de datos, esto facilitaría un poco las funciones a desarrollar por el usuario y el manejo de los programas (primitivas, revolución, etc.)

Interfaz gráfica: el manejo de los objetos a través de la cuadrícula de Linux, permite sólo la graficación de un (1) objeto, se pueden tener 2 objetos en una misma cuadrícula, pero serían la consecuencia de una operación booleana (por lo que sería un solo objeto y no 2 como se dijo anteriormente), se propone ampliar dicha limitante para entonces tener en la representación gráfica n-objetos de interés para su visualización por parte del usuario.

Captura de pantalla: Los objetos aquí mostrados están de alguna manera generados a partir de las funciones de creación de primitivas, dichas primitivas son creadas de una única manera y en una única posición, por lo que se hace imposible si un cono que fue creado con el eje de simetría paralelo a X, sea almacenado con el eje de simetría paralelo a Y o a Z, la opción de capturar pantalla permitiría a un usuario graficar un objeto, rotarlo en cualquiera de las direcciones (X Y Z) y al activar la opción de capturar pantalla, ésta relacione las posiciones de los vértices trasladados para la creación de un nuevo archivo de información.

Sombreado y vistas: el Sistema de Modelador dé una visualización del objeto en el espacio, pero queda mucho a la perspectiva del usuario la posición de dicho sólido, esto luego se puede solventar si desde un punto de vista se indican las aristas ocultas o las sombras de las caras a partir de cierto foco.

Herramientas de medición: los objetos planteados en la cuadrícula de visualización si bien están creados en función de un patrón de medición, sería útil representar una escala de graficación o incluso si el usuario señalara 2 puntos cualesquiera sobre el sólido, la función retorne el valor de la distancia que existe entre ellos (por su puesto es necesaria la captura de video para poder desarrollar dicha aplicación).

Ampliar las capacidades: para poder manipular elementos más complejos las capacidades de dicho programa deben ser ampliadas, por ello no solo son las constantes sino la verificación de los procesos lo que conlleva a dicho estudio y posterior consideración

APENDICE A

MANUAL DE USUARIO

Instalando la aplicación:

Para iniciar la aplicación, es necesario copiar la carpeta “Programa_SMS” en el directorio raíz de LINUX, esta carpeta contiene tanto los ejecutables como ejemplos de sólidos que sirven para introducir al usuario en el manejo de la aplicación.

Para un buen desempeño de la aplicación, es recomendable editar el archivo “.profile” y colocar el siguiente escrito al final del mismo “`PATH=${PATH}:.`” Esto hará que la aplicación se pueda ejecutar sin tener que escribir “./ “ antes de cada comando (requisito de LINUX para la ejecución de aplicaciones).

Una vez instalado el programa, se puede ejecutar con simplemente escribir “programa_sms”.

Nota: si se escribe “programa_sms Nombrearchivo.res” el programa automáticamente grafica el archivo indicado, esto pudiera ser de utilidad para algunas aplicaciones, como lo son la vista rápida de archivos.

Manejo de la aplicación:

Una vez que se ha ejecutado el programa, se pueden crear sólidos con los distintos botones que dispone, la explicación de los mismos se encuentra en ayuda del programa, o en Apéndice B del presente Trabajo Especial de Grado.

El directorio raíz desde el cual trabaja el programa es el indicado y por lo tanto todos los sólidos a ser manipulados deben ser ubicados en dicha carpeta. Así mismo los sólidos que se generen por los procesos del programa estarán en esa ubicación.

Es importante recalcar que hay que tener cuidado con los nombres de los archivos y los procesos que se están aplicando ya que si erróneamente se escribe el nombre de un archivo existen en el proceso de crear un archivo nuevo, éste automáticamente borrará el archivo anterior para dar paso al nuevo archivo.

El sistema operativo LINUX es sensible a las mayúsculas por lo que es importante tomar en cuenta a la hora de asignar los nombres para luego al hacer el llamado pueda hacerse de manera fácil y rápida.

Módulo de operaciones Booleanas:

El Módulo posee un formato muy similar a las demás aplicaciones, al ingresar se desplegará el menú de operaciones a realizar……

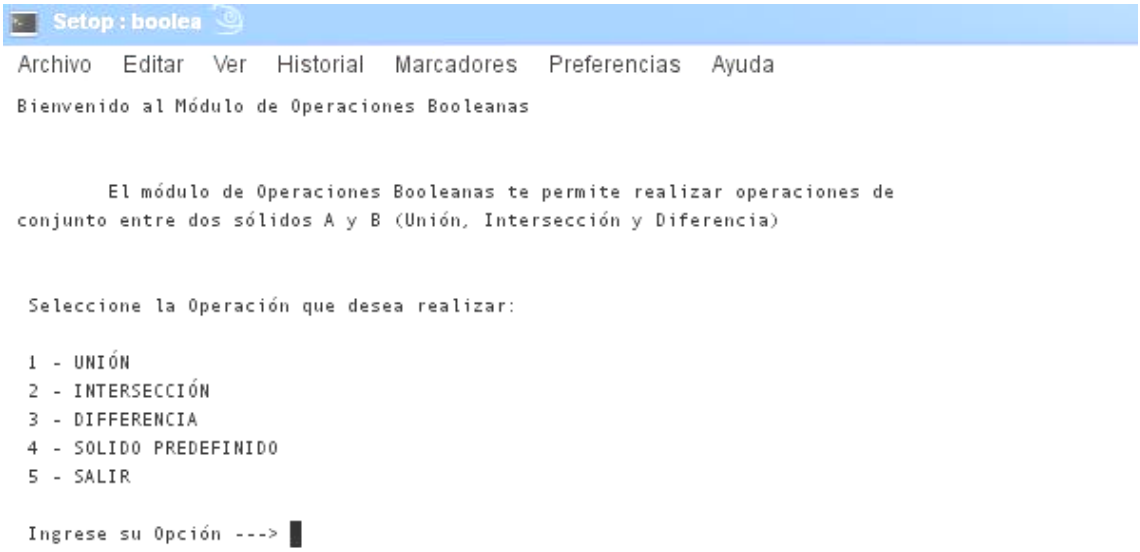


Figura 80. Menú Operaciones Booleanas

Para dicho menú hay que indicar que los operadores determinan el sólido de mayor cantidad de vértice y lo toman como el elemento base, para explicar esto:

Sean Sol1 y Sol2 dos sólidos cualesquiera en los que Vert1 > Vert2, las operaciones se darán como sigue:

	Operación	Resultado
Sol1 = A Sol2 = B	1	Intersección
Sol1 = A Sol2 = B	2	Unión
Sol1 = A Sol2 = B	3	Diferencia (Sol2 - Sol1)

Figura 81 Listado Operaciones Booleanas

Si en cambio se invirtiera el orden de entrada de datos se obtendría lo siguiente para las mismas operaciones:

	Operación	Resultado
Sol2 = A Sol1 = B	1	Diferencia (Sol2 - Sol1)
Sol2 = A Sol1 = B	2	Diferencia (Sol1 - Sol2)
Sol2 = A Sol1 = B	3	Intersección

Figura 82 Listado Operaciones Booleanas (continuación).

Se presenta como cuarta selección la demostración de un sólido ejemplo (predefinido) para la Operación Booleana:

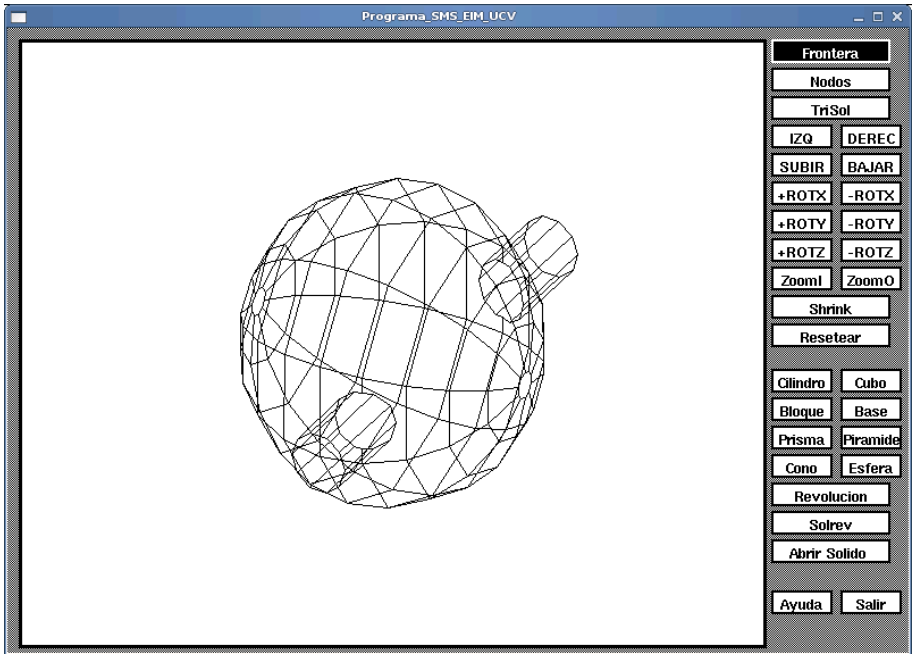


Figura 83 Ejemplo predefinido Operación Booleana

Luego el usuario deberá indicar el nombre del archivo con que se guardará el resultado de la operación booleana, el esquema como presenta el programa se muestra a continuación en la figura.

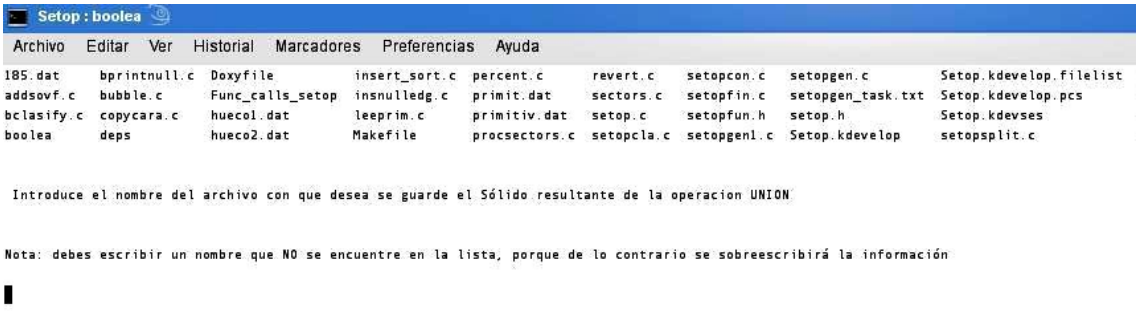


Figura.84 Menú de Selección Resultado Operación Booleana

Es muy importante resaltar la observación que aparece al final del listado "Nota: debes escribir un nombre que NO se encuentre en la lista, porque de lo contrario se sobreescribirá la información"

La razón de ser de dicha advertencia es la del comando de creación de dicho archivo; si el archivo no existe, la función lo crea pero si el archivo ya previamente existe, la función de crear el archivo lo inicializa (borra su contenido) y lo prepara para escribir los datos nuevos

Posteriormente se le solicitará al usuario que coloque los nombres de los archivos de los sólidos desea operar como A y B, el menú se presenta a continuación en la figura

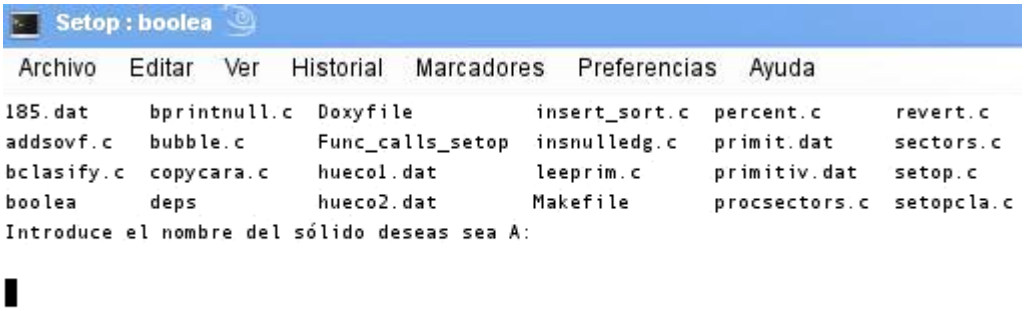


Figura 85 Menú de Selección de Sólidos A y B.

El programa aquí presentado desarrollado bajo los parámetros definidos por Mäntylä pudiera presentar alguna incongruencia en el manejo de los sólidos más complejos de lo que aquí se muestra, esto es debido a los parámetros máximos y las capacidades, se plantea dentro de las recomendaciones del Trabajo Especial de Grado. Entre las limitantes están: el número de caras, el número de caras nulas, la cantidad de vértices, la cantidad de nodos, las dimensiones de los objetos, entre otros.

Los parámetros de diseño del programa permanecen ocultos a la vista de los usuarios por lo que para su modificación debe realizarse en manos de personas con conocimientos de programación.

Compilación del Programa:

La estructura del programa esta desarrolla con Make / Makefile que es un tipo de arreglo de proyecto, esto permite la compilación por partes del programa (quiere decir que si se modifica una función, no hay necesidad de compilar el programa

completo sino sólo la función que fue modificada); la estructura de los Makefile arroja el ejecutable del programa en la carpeta “bin” del sistema y los ejecutables de las funciones en la carpeta “Test” exceptuando el Módulo de aplicaciones Booleanas que se podrán encontrar en la carpeta “Setop”.

Todos los ejecutables deberán ser ubicados en un mismo subdirectorio para que la interfaz gráfica pueda enlazar las funciones con los botones que así lo requieren, por otro lado la función voltear que se encuentra ubicada en la carpeta “Test” aparece el ejecutable en la misma ruta, y se puede llamar con tipear “voltear nombrearchivo.res”

Portabilidad:

Para llevar el programa a otro computador se recomienda desde el directorio raíz del código fuente escribir “make backup”, este comando generará un archivo “.tgz” llamado “TesisErnesto.tgz” que contendrá el programa, al llegar al nuevo computador, sólo se debe descomprimir dicho archivo en una carpeta que se llame Programa_SMS

APENDICE B

AYUDA DEL PROGRAMA SMS_EIM_UCV

DESCRIPCIÓN DEL PROGRAMA: Sistema de Modelado de Sólidos que permite la creación automatizada y la manipulación de modelos tridimensionales de piezas complejas mediante Interacción Gráfica.

AUTOR: Éste programa ha sido elaborado por el Bachiller Laurent Chevron como parte de su Trabajo Especial de Grado para optar por el título de Ingeniero Mecánico ante la ilustre Universidad Central de Venezuela, bajo la tutoria del Prof. Antonio Barragán, posteriormente sus capacidades fueron ampliadas como parte del Trabajo Especial de Grado para optar por e l título de Ingeniero Mecánico, ante la ilustr Universidad Central de Venezuela por el Bachiller Ernesto Centeno, igualmente bajo la tutoría del Prof. Antonio Barragán.

Gracias por Usar Software libre y 100% UCV!!!

	BOTON	FUNCIÓN
CREAR SÓLIDOS	BOOLEA	Ejecuta el Módulo de Operaciones Booleanas.
	CILINDRO	Ejecuta programa para generar cilindros.
	CUBO	Ejecuta programa para generar cubos.
	BLOQUE	Ejecuta programa para generar bloques de dimensiones ancho, largo, altura.
	BASE	Ejecuta programa para generar sólidos por extrusión a partir de una base.
	PRISMA	Ejecuta programa para generar prismas rectos cuya base es un polígono.
	PIRÁMIDE	Ejecuta programa para pirámides rectas.
	CONO	Ejecuta programa para generar conos.
	ESFERA	Ejecuta programa para generar esferas.
	REVOLUCION	Ejecuta programa para generar sólidos de revolución macizos .
	SOLREV	Ejecuta programa para generar sólidos de revolución huecos.
	ABRIRSOLIDO	Permite ver en pantalla un sólido creado anteriormente.

	BOTON	FUNCIÓN
MANIPULAR SÓLIDOS	FRONTERA	Permite quitar/poner aristas de un sólido en la pantalla.
	NODOS	Permite quitar/poner vértices de un sólido en la pantalla.
	IZQ	Permite desplazar el sólido hacia la izquierda de la pantalla.
	DEREC	Permite desplazar el sólido hacia la derecha de la pantalla.
	SUBIR	Permite desplazar el sólido hacia arriba de la pantalla.
	BAJAR	Permite desplazar el sólido abajo de la pantalla.
	+ROTX	Permite girar el sólido alrededor del eje X (regla mano derecha).
	+ROTY	Permite girar el sólido alrededor del eje Y (regla mano derecha).
	+ROTZ	Permite girar el sólido alrededor del eje Z (regla mano derecha).
	ZOOM I	Permite realizar un acercamiento del sólido.
	ZOOM O	Permite realizar un alejamiento del sólido.
	RESETEAR	Permite regresar un sólido a su tamaño original.
	AYUDA	Permite ver las funciones de cada boton.
	-ROTX	Permite girar el sólido alrededor del eje X (regla mano izquierda).
	-ROTY	Permite girar el sólido alrededor del eje Y (regla mano izquierda).
	-ROTZ	Permite girar el sólido alrededor del eje Z (regla mano izquierda).

APÉNDICE C

FUNCIONES Y UBICACIÓN

Las funciones implementadas y descritas en el programa están ubicadas de forma ordenada en varios de los archivos incluidos en el código fuente del programa, su uso de los archivos Setop y Testsetop están indicados, estas funciones pueden ser ubicadas según el directorio que se muestra a continuación.

Funciones TODAS

	carpeta	testsetop.c	setop.c	archivo
addhe	lists	x		addhelhe.c
AddListEdge	lists	x		Listaddel.c
AddListFace	lists	x	x	Listaddel.c
AddListLoop	lists	x	x	Listaddel.c
AddListVert	lists	x		Listaddel.c
addsovf	Setop		x	addsovf.c
addsovv	Setop		x	addsovf.c
AddVeF2VList	Setop		x	addsovf.c
AddVeV2VList	Setop		x	addsovf.c
arc	utils	x		primitiv.c
Ball	utils	x		primitiv.c
bMueveCaras	Setop		x	bclasify.c
BorraEdge	lists		x	allocmem.c
BorraLoop	Lists		x	allocmem.c
BorraHalfEdge	Lists		x	allocmem.c
bSplitNullFaces	Setop		x	setopfin.c

Funciones TODAS

	carpeta	testsetop.c	setop.c	archivo
BuildVert	Utils	x		buildvert.c
CheckBounds	Utils		X	checbound.c
Cilindro	Utils	x		primitiv.c
comp	Utils		x	vector.c
CreaEdge	Utils	x		allocmem.c
CreaFace	Lists	x		allocmem.c
CreaHalfedge	Lists	x		allocmem.c
CreaLoop	Lists	x	x	allocmem.c
CreaSolid	Lists	x	x	allocmem.c
CreaVert	Lists	x		allocmem.c
cutab	Setop		x	setopsplit.c
delhe	Lists		x	addelhe.c
DelListEdge	Lists		x	Listaddel.c
DelListLoop	Lists		x	Listaddel.c
DelListFace	Lists		x	Listaddel.c
doSetOpGenerate	Setop		x	setopgen.c
DoVertexOnFace	Setop		x	setopgen.c
dropcoord	Utils		x	contain.c
Fac2Vert	Utils	x		Fac2Vert.c
faceeq	Utils	x	x	opgeom.c
fface	Utils	x		scanc.c
GetAxesOfFace	Utils	x		opgeom.c
GetAxis	Utils	x		opgeom.c
InOppSides	Setop		X	setopgen.c
InsertNullEdges	Setop		x	Insnulledg.c
join	Setop		x	setopsplit.c

Funciones TODAS

	carpeta	testsetop.c	setop.c	archivo
KillProgram	Utils	x	X	Killprog.c
MatIdent	utils	x		vector.c
Matrotat	utils	x		vector.c
movefac	Setop		x	bclasify.c
myAllocMem	lists	x	x	allocmem.c
OP_EU_lkemr	euler		x	leulerop.c
OP_EU_lmef	euler	x		leulerop.c
OP_EU_lmekr	euler		x	leulerop.c
OP_EU_lmev	euler	x	x	leulerop.c
OP_EU_mef	euler	x		eulerop.c
OP_EU_mvfs	euler	x		leulerop.c
PrintAdyaVE	utils	x		listsoli.c
PrintEdges	utils	x		listsoli.c
PrintFace	utils	x		listsoli.c
PrintFaceList	utils	x		listsoli.c
PrintLoop	utils	x		listsoli.c
PrintVertList	utils	x		listsoli.c
PrintVertNormals	utils	x		listsoli.c
process_edge	Setop		x	setopgen.c
ScaleVec	Utils	x		vector.c
ScaleVect	Utils	x		vector.c
scanjoin	setop		x	setopcon.c
sgetnextnulledge	Setop		x	setopcon.c
sgreater	Setop		x	setopsplit.c
SetOpClasiffy	Setop	x		setopcla.c
SetOpConnect	Setop	x		setopcon.c

Funciones TODAS

	carpeta	testsetop.c	setop.c	archivo
SetOpFinish	Setop		x	setopfin.c
SetOpGenerate	Setop		x	setopgen.c
Solidls	utils	x		listsoli.c
ssortnulledges	Setop		x	setopcon.c
sswap	Setop		x	setopcon.c
Sweep	utils	x		primitiv.c
sweepR	utils	x		primitiv.c
VecCopy	utils	x		vector.c
VecMult	utils	x		vector.c
Vecplus	utils	x		vector.c

Glosario de Términos

CAD: Asistente Computarizado de Diseño en inglés las siglas significan “Computer Aided Desing”

CAE: Asistente Computarizado Ingenieril, en inglés “Computer Aided Engineering”

CAT: Asistente Computarizado de Pruebas, en inglés “Computer Aided Testing”

CAM: Asistente Computarizado de Manufactura, en inglés “Computer Aided Manufacturing”

CAPP: Asistente Computarizado de Planificación y Procesamiento “Computer Aided Processing and Planning”

Pixel: O también conocidos como Pels (elementos de imágenes o picture elements, en inglés)

DEFINICIONES

2 - Manifold:

También traducido como 2 – variedades, se refiere al término que interpreta cuando una arista pertenece a 2 caras de un mismo sólido. Cada punto tiene una vecindad de puntos que lo rodean y forman un disco topológico (a) y (b). Si un objeto no es una 2-variedad, entonces no tiene puntos cuya vecindad sea un disco topológico.

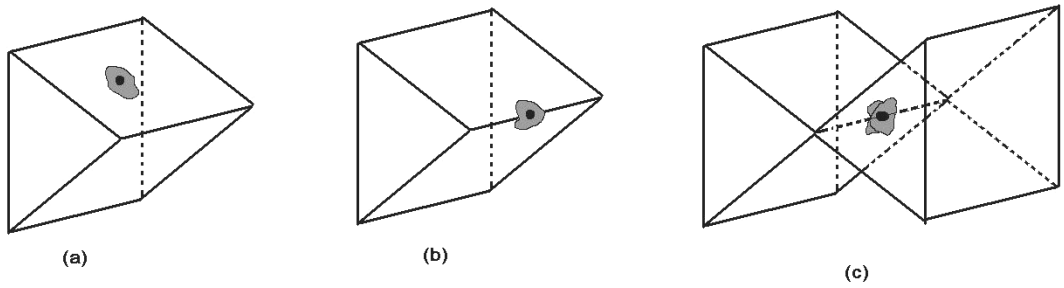


Figura 1. Descripción 2 – variedades. [FOLE94]

En los casos de operaciones booleanas es de suma importancia determinar si un objeto es de 2 variedad o no ya que de ello depende el manejo de ciertos operadores, el hecho ocurre porque se puede estar en presencia de 2 objetos que son 2-variedades (a) y obtener como resultado un objeto (b) que no lo es, se puede apreciar a continuación en la Figura 2.

B-Spline:

El término *Spline* proviene de las largas tiras flexibles de metal que usaban los Dibujantes para establecer las superficies de aeroplanos, automóviles y barcos. A las *Spline* se le colocaban pesos que servían para halar la *Spline* en varias direcciones. A menos que se sometieran a grandes tensiones, las *Spline* metálicas tenían continuidad de segundo orden. El equivalente matemático de estas tiras, La *Spline* cúbica natural, es un polinomio cúbico con continuidad C0, C1 y C2 que interpola (pasa por) los puntos de control. Este polinomio tiene un grado de continuidad más que la continuidad inherente en las formas de Hermite y Bézier. Por lo tanto, las *Spline* son más suaves que las formas anteriores, en algunas referencias se trata la B-spline como la forma resumida de Bézier, a continuación se muestran unos ejemplos:

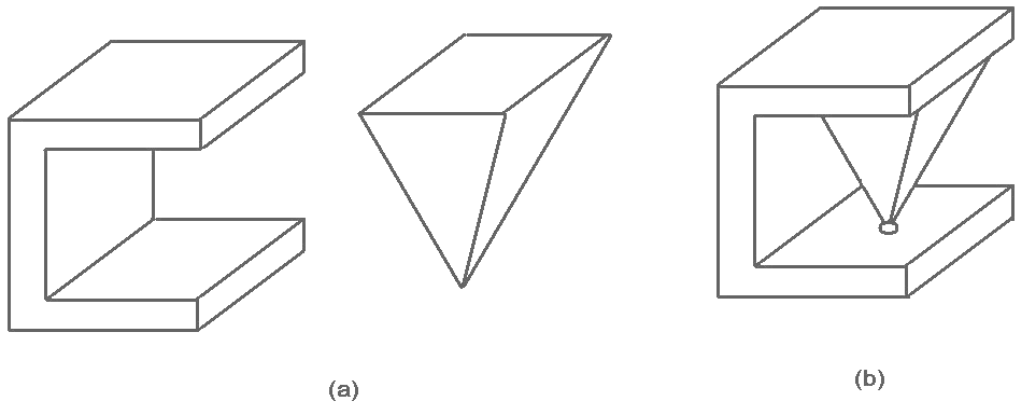


Figura 2. Operación booleana de 2 objetos 2- variedad generando un objeto no variedad. [WEIL88]

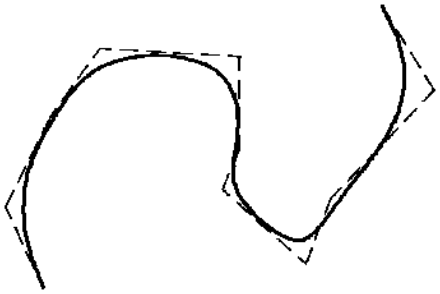


Figura 3. Suavizado mediante B-spline (I6)

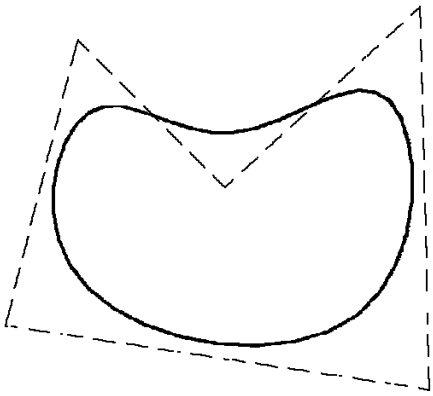


Figura 4. Suavizado de un polígono mediante curvas de B splines cúbicas (I6).

`For`: Palabra reservada del Lenguaje de Programación “C” y se utiliza para crear lazos iterativos, se define como se indica a continuación:

```
For (inicialización; condición; incremento)
{ACCIÓN A REALIZAR};
```

Inicialización: el usuario colocará el valor inicial de la variable

Condición: Será el criterio de parada para el lazo indicado.

Incremento: será la variación que considerará la variable cada vez que se cumpla un lazo

Gráficos Raster:

También llamada Bitmap, imagen matricial o pixmap, es una estructura o fichero de datos que representa una rejilla rectangular de pixeles o puntos de color, denominada Raster que se puede visualizar en un dispositivo gráfico.

Los gráficos rasterizados se distinguen de los gráficos vectoriales en que estos últimos representan una imagen a través del uso de objetos geométricos como curvas de Bézier y polígonos, no del simple almacenamiento del color de cada pixel. El formato de imagen matricial está ampliamente extendido y es el que se suele emplear para tomar fotografías digitales y realizar capturas de vídeo. Para su obtención se usan dispositivos de conversión analógica-digital, tales como escáneres y cámaras digitales.

La calidad de las imágenes rasterizadas es determinada por el total de píxeles que poseen (resolución) y la cantidad de información en cada píxel (generalmente llamada profundidad de color). Por ejemplo, una imagen que almacena 24 bits de información por cada píxel (que es el estándar para todas las pantallas desde 1995) puede representar de forma más suave las tonalidades que una imagen que almacena sólo 16 bits por píxel.

De igual manera en cuanto a la resolución, una imagen de 640 x 480 píxeles (con un total de 307.200 píxeles constituyentes) se verá más accidentada y pixelizada comparada con una de 1280 x 1024 (que posee 1.310.720 píxeles constituyentes), no es recomendable ampliar una imagen de este tipo ya que la información que almacena

el archivo de imagen, es a la resolución que fue creada, si se ampliase, carecería de múltiples parámetros, por lo que se vería seriamente afectada. Como se observa a continuación.

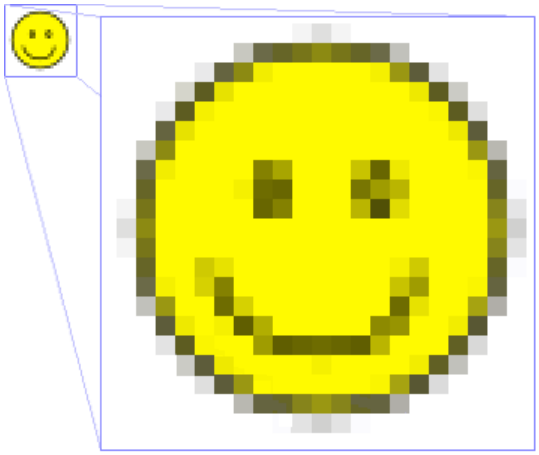


Figura 5 Ejemplo de Imagen Rasterizada (I5)

Non Manifold:

Este término también esta asociado a “Singularidad”, en la imagen planteada a continuación, se puede apreciar dos objetos completamente normales, definidos (a) cuya estructura es del tipo manifold, luego si estos objetos se unen (b) se puede generar el caso de una estructura Non Manifold.

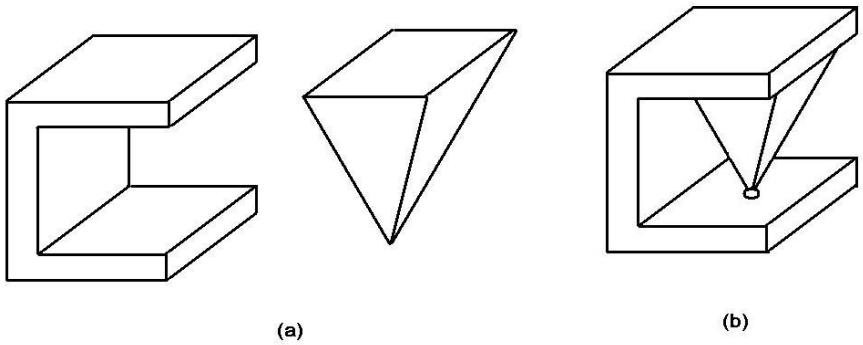


Figura 6 Objeto Non Manifold (WEIL88)

La estructura de objeto non manifold carece de funcionabilidad, por lo que la aplicación al menos en el campo de ingeniería es sin sentido.

Singularidad:

La Singularidad, dentro de la generación de objetos es muy importante estudiarla, en el caso de las aplicaciones por siendo lo mas importante la idea de Generar objetos no Realizables, es de destacar que en la literatura se pueden apreciar Trabajos de Escher (Maurits Cornelis Escher 1898-1972), quien realizó múltiples ensayos sobre objetos imposibles, a continuación se muestran algunos de ellos:

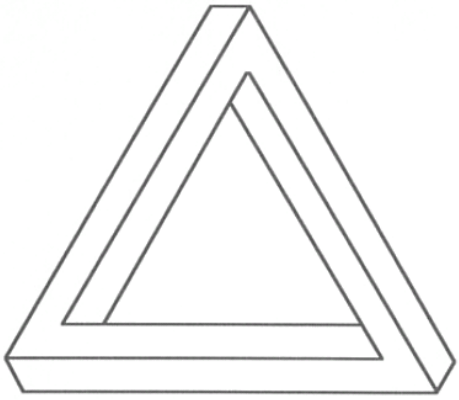


Figura 7. Obra “Penrose Triangle” (I7)

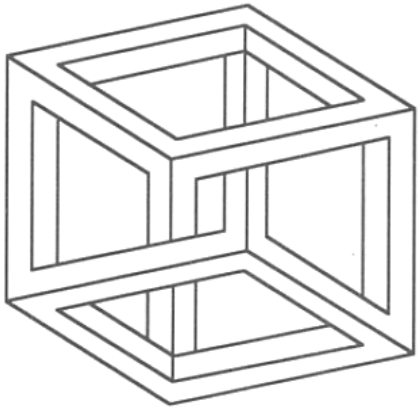
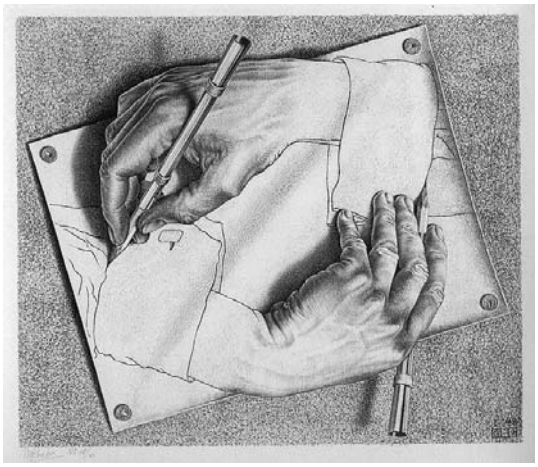


Figura 8. Obra “Escher Cube” (I7)

Las Figuras 7 y 8 pueden ser tomadas en cuenta como ejemplos de la ambigüedad por su planteamiento de los vértices si se analiza el inicio y el final de los mismos.

Entre las obras famosas mas emblemáticas de éste artista, se encuentra “Drawing Hands” donde se aprecia claramente el detalle de la pintura.



A simple vista parecen ser pinturas o paisajes comunes, pero si es observado con detalle y raciocinio, se puede observar que hay contradicciones de diseño, que dan entonces poco sentido a la lógica estructural.

Figura 9. Obra “Drawing Hands” (1948) (I9).

REFERENCIAS BIBLIOGRÁFICAS

CHEV08 Chevron, L. 2008. Implementación de un Sistema de Modelado de Sólidos para la creación de Primitivas Geométricas e interacción Gráfica, con Aplicaciones en la Ingeniería Mecánica. TESIS. Universidad Central de Venezuela.

FOLE94 Foley, J. 1994. Introducción a la graficación por computador, editorial Addison-Wesley Iberoamericana.

JAIM09 Jaimes, J. Jauregui, M. Expansión de las Capacidades de un Programa de Visualización de Resultados para Programa de Simulación Numérica Mediante el Método de Elementos Finitos. TESIS. Universidad Central de Venezuela

MANT 88 Mäntyla, M. 1988. Introduction to Solid Modeling, Computer Science Press, Rockville, Maryland.

MARC96 Marcheix, D 1996. Modelisation géometrique des objets non-variétés représentation, construction et déformation. Thése. Université Bordeaux.

MORT85 Mortenson, M. 1985. Geometric Modeling. Editorial Willey, Nueva York.

UZCA93 Uzcátegui, J. Uzcátegui, C. 1993. Implementación de técnicas básicas de modelado de sólidos con implementación en la Ingeniería Mecánica. TESIS Universidad Central de Venezuela.

WEIL88 Weiler, K.1988. The Radial Edge Structure: A Topological Representation for Non – Manifold Geometric Boundary Modeling. ABSTRACT.

WEIL88(2)Weiler, K. 1988. Boundary Graph Operators for Non – Manifold Geometric Modeling Topology Representations. ABSTRACT.

INTERNET

1)http://www.unizar.es/aeipro/finder/INGENIERIA%20DE%20PRODUCTOS/BF04..htm

Consultado: Lunes 16 de febrero de 2008.

2)*lim.ii.udc.es/docencia/cax2002/CAX2002_intro.ppt*

Consultado: Lunes 16 de febrero de 2008.

3) http://mrl.nyu.edu/publications/nonmanifold/nonmanifold.pdf

Consultado: Lunes 16 de febrero de 2008.

4) http://www.alegsa.com.ar/Dic/grafico%20rasterizado.php

Consultado: jueves 26 de marzo de 2009

5) http://es.wikipedia.org/wiki/Archivo:Rgb-raster-image.png

Consultado: jueves 26 de marzo de 2009

6) http://www6.uniovi.es/mieres/egi/dao/apuntes/poligonos_interpolantes.html

Consultado: jueves 26 de marzo de 2009

7) http://dac.escet.urjc.es/docencia/IG/02-ModeladoGeometrico.pdf

Consultado: jueves 26 de marzo de 2009

8) http://ignorantisimo.free.fr/3D/docs/3D2005_APT01_-_Capitulo_I_Representacion_por_poligonos.pdf

Consultado: jueves 26 de marzo de 2009

9) http://en.wikipedia.org/wiki/M._C._Escher

Consultado: jueves 26 de marzo de 2009