UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

TRABAJO DE GRADO

OPERACIONES DE CÓMPUTO GRÁFICO EN EL ESPACIO GEOMÉTRICO CONFORME 5D USANDO GPU

por

Lic. Eduardo Roa

Sartenejas, Mayo de 2011

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

OPERACIONES DE CÓMPUTO GRÁFICO EN EL ESPACIO GEOMÉTRICO CONFORME 5D USANDO GPU

Trabajo Especial de Grado presentado a la Universidad Simón Bolívar por

Lic. Eduardo Roa

como requisito parcial para optar al grado **académico** de

Maestría en Ciencias de la Computación

Con la asesoría del profesor:
Victor Theoktisto

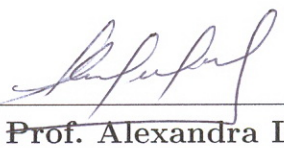
Sartenejas, Mayo de 2011

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

OPERACIONES DE CÓMPUTO GRÁFICO EN EL ESPACIO GEOMÉTRICO CONFORME 5D USANDO GPU

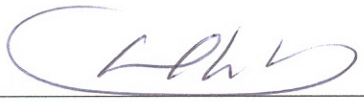
Por: Lic. Roa Marino, Eduardo
Carnet Nro. 0786219

Este **Trabajo Especial de Grado** ha sido aprobado en nombre de la Universidad Simón Bolívar por el siguiente jurado examinador:



Prof. Alexandra La Cruz
Presidente

Prof. Rhadames Carmona
Miembro Principal
Universidad Central de Venezuela



Prof. Victor Theoktisto
Miembro Principal - Tutor

Sartenejas, Viernes 06 de Mayo de 2011.

AGRADECIMIENTOS

A Dios por haber estado conmigo en los momentos mas díficiles en la carrera.

A mis padres, parte esencial e importante en mi vida, de quién solo he recibido apoyo y motivación para el logro de mis ideales y metas. Sin ellos no hubiese logrado lo que soy hoy.

A los profesores Victor Theoktisto y Alexandra la Cruz quienes fueron mis mentores en el área de gráficas.

A los profesores Francisco Tovar y Marco Paluszny por su gran amistad y ayuda cuando la he necesitado.

A todos mis compañeros y amigos con quienes he compartido momentos muy emotivos, alegres, y que siempre me han apoyado.

UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

OPERACIONES DE CÓMPUTO GRÁFICO EN EL ESPACIO GEOMÉTRICO CONFORME 5D USANDO GPU

Por: Lic. Roa Marino, Eduardo
Carnet Nro. 0786219

RESUMEN

La geometría conforme permite una representación simple de las primitivas que se usan con frecuencia en la computación gráfica, como lo son puntos, planos y esferas. Además, este modelo conforme representado en 5D presenta una sola fórmula para cubrir todas las intersecciones de las primitivas antes mencionadas.

El modelo conforme usa como marco de trabajo el álgebra geométrica, que permite una simplificación de ciertas operaciones como calcular el área de un paralelogramo que está representado por un bivector o el cálculo de un volumen representado por un trivector.

El objetivo de este trabajo es trabajar con estas primitivas en un área de la computación gráfica que son las colisiones e intersecciones. Para optimizar el tiempo de cómputo, se pretende realizar el cálculo en programación paralela usando el GPU (Graphics Processor Unit). Por último, no se utilizan volúmenes delimitadores de colisión para optimizar los algoritmos; las colisiones se harán en el peor de los casos, triángulo a triángulo, de tal manera de observar los tiempos de cómputo sin hacer una simplificación en los datos.

Palabras Claves: Geometría Conforme, Cálculo de Colisiones, Algebra Geométrica, CUDA, Computación Paralela.

ÍNDICE GENERAL

APROBACIÓN DEL JURADO ii

AGRADECIMIENTOS iii

RESUMEN iv

ÍNDICE GENERAL v

ÍNDICE DE FIGURAS viii

GLOSARIO x

INTRODUCCIÓN 1

I. ÁLGEBRA GEOMÉTRICA 3
1. Producto Externo (Outer Product) . . . 3
   1.1. Producto externo en 2D y 3D . . . 5
2. Producto Geométrico (Geometric Product) . . . 6
   2.1. Producto Geométrico de la base . . . 8
   2.2. Propiedad imaginaria . . . 8
   2.3. Trivectores . . . 9
3. Dualidad . . . 12
4. Reflexión y Rotaciones . . . 13
   4.1. Reflexión . . . 13
   4.2. Rotación . . . 14
5. Algunas aplicaciones del álgebra geométrica . . . 15
   5.1. Punto dentro de un triángulo . . . 15
   5.2. Orientación de un punto con relación a una recta . . . 16
   5.3. Orientación de un punto con relación a un plano . . . 17
   5.4. Distancia más corta de un punto a un plano . . . 18

II. GEOMETRÍA CONFORME 20
1. Modelo Conforme . . . 21
   1.1. Elementos del modelo conforme . . . 22
   1.2. Punto . . . 23
   1.3. Recta . . . 24
   1.4. Círculo . . . 25
   1.5. Plano . . . 25
   1.6. Esfera . . . 26
2. Transformaciones . . . 27

2.1. Traslación . . . 27
2.2. Rotación . . . 28
2.3. Dilatación (Escalamiento o Escalado) . . . 29
2.4. Reflexión . . . 29
3. Intersecciones de primitivas . . . 30
3.1. Intersección Recta-Recta . . . 30
3.2. Intersección Recta-Plano . . . 31
3.3. Intersección Recta-Esfera . . . 32
3.4. Intersección Plano-Plano . . . 34
3.5. Intersección Esfera-Plano . . . 34
3.6. Intersección Esfera-Esfera . . . 36

III.COMPUTACIÓN PARALELA Y COLISIONES 38
1. Computación Paralela . . . 38
1.1. Historia . . . 38
2. GPU (Graphics Processor Unit) . . . 40
2.1. Vertex Shader y Fragment/Pixel Shader . . . 43
2.2. Paralelismo en GPU . . . 45
2.3. CUDA - Compute Unified Device Architecture . . . 48
2.4. CPU vs GPU . . . 53
3. Intersección y Colisiones . . . 54

IV.IMPLEMENTACIÓN 59
1. Algoritmos . . . 59
1.1. Intersección Segmento de Recta-Segmento de Recta . . . 59
1.2. Intersección Segmento de Recta-Triángulo . . . 61
1.3. Intersección Segmento de Recta-Esfera . . . 63
1.4. Intersección Triángulo-Triángulo . . . 65
1.5. Intersección Triángulo-Esfera . . . 66
1.6. Intersección Esfera-Esfera . . . 69
2. Implementación . . . 70
2.1. Programa . . . 71
2.2. Configuración CUDA . . . 72
2.3. Detalles de implementación . . . 72
2.4. Tiempo de ejecución . . . 74

V. RESULTADOS 75
1. Intersección de primitivas con una malla poligonal . . . 75
1.1. Colisión de dos mallas poligonales . . . 76
1.2. Colisión de Esferas con primitivas . . . 76
1.3. Análisis . . . 78
2. Geometría Euclidea 3D vs Geometría Conforme 5D . . . 81

VI.CONCLUSIÓN 83

A. INTERSECCIÓN Y TRANSFORMACIONES **84**
1. Primitivas 84
1.1. Recta 84
1.2. Plano 86
1.3. Esfera 89
2. Intersecciones 91
2.1. Recta-Recta 91
2.2. Recta-Plano 93
2.3. Recta-Esfera 94
2.4. Plano-Plano 95
2.5. Esfera-Plano 95
2.6. Esfera-Esfera 96
3. Rotación 97
4. Reflexión 99

B. CAPTURAS DE PANTALLA **103**

REFERENCIAS BIBLIOGRÁFICAS **107**

ÍNDICE DE FIGURAS

I.1. Orientación de un bivector 4
I.2. Áreas proyectadas de un bivector 6
I.3. Volumen de un trivector 10
I.4. Reflexión de un vector 13
I.5. Rotación de un vector 14
I.6. Punto fuera de un triángulo 15
I.7. Punto dentro de un triángulo 16
I.8. Orientación de un punto con respecto a una recta 17
I.9. Orientación de un punto con respecto a un plano A 17
I.10. Corta distancia de un punto a un plano 18

III.1. Tecnología Hyperthread [16] 39
III.2. Tecnología DualCore [17] 40
III.3. Sillicon Graphics - Indy [19] 41
III.4. SIMD (Single instruction, multiple data) 42
III.5. Id-Software - Quake [25] 42
III.6. Pipeline gráfico [26] 43
III.7. Vertex Shader y Pixel Shader Pipeline [26] 44
III.8. Funcionamiento del Vertex Shader y Pixel Shader 45
III.9. Relieve - Bump Mapping [29] 46
III.10. Campo Profundidad - Depth of Field [30] 46
III.11. Paralelismo en GPU 47
III.12. CUDA: Grid, Bloque e Hilo 49
III.13. Memoria en CUDA 50
III.14. GigaFlops del CPU y GPU [42] 53
III.15. BandWidth/seg del CPU y GPU [42] 54
III.16. Colisiones - Malla deformable [43] 54
III.17. Caja de borde alineada - AABB 55
III.18. Esfera 56
III.19. Caja Orientada - OBB 56
III.20. Cápsula Convexa, Unreal Development Kit [50] 57
III.21. *Binary space-partitioning* 57

IV.1. Intersección segmento de recta - segmento de recta 60
IV.2. Intersección segmento de recta - triángulo 63
IV.3. Intersección segmento de recta - esfera 64
IV.4. Intersección Triángulo-Triángulo 66
IV.5. Casos posibles en intersección Triángulo-Esfera 68
IV.6. Intersección Triangulo-Esfera 69
IV.7. Autodesk Maya 70

IV.8. Interfaz del Programa 71
IV.9. Normalizar puntos de la recta 73

V.1. Intersección Segmento - Malla Bunny 75
V.2. Intersección Triángulo - Malla Bunny 76
V.3. Intersección Esfera - Malla Bunny 77
V.4. Intersección Malla Bunny - Malla Armadillo 78
V.5. Intersección Esferas - Segmento Recta 79
V.6. Intersección Esferas - Triángulo 80
V.7. Intersección Esferas - Esferas 80
V.8. Intersección Malla Bunny - Segmento Recta 81
V.9. Intersección Malla Bunny - Triángulo 82

B.1. Intersección Malla Bunny - Segmento de Recta 103
B.2. Intersección Malla Bunny - Triángulo 104
B.3. Intersección Malla Bunny - Esfera 104
B.4. Intersección Malla Bunny - Malla Armadiillo 105
B.5. Intersección Esferas - Triángulo 105
B.6. Intersección Esferas - Segmento de Recta 106
B.7. Intersección Esferas - Esferas 106

GLOSARIO

BSP Partición Binaria del Espacio (BSP) es un método para subdividir recursivamente un espacio en elementos convexos empleando hiperplanos. Esta subdivisión da lugar a una representación de la escena por medio de una estructura de datos del árbol conocida como árbol de BSP.

Bump Mapping es una técnica de gráficos computacionales 3D creada por James F. Blinn en 1978. Consiste en dar un aspecto rugoso a las superficies de los objetos. Esta técnica modifica las normales de la superficie sin cambiar su geometría.

Clipping es el proceso de cortar los triángulos de manera que ocupen el área visible.

CPU La unidad central de procesamiento o CPU (por el acrónimo en inglés de central processing unit), o simplemente el procesador o microprocesador, es el componente del computador y otros dispositivos programables, que interpreta las instrucciones contenidas en los programas y procesa los datos.

Depth of Field se entiende tradicionalmente en óptica, y en fotografía en particular, como la zona en la cual la imagen captada por el objetivo es nítida (es decir enfocada), de manera que en la fotografía que se realice, las personas y objetos que se encuentren dentro de esa zona aparecerán también nítidos.

DIRECTX DirectX es una colección de API desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

GPU La unidad de procesamiento gráfico o GPU (acrónimo del inglés graphics processing unit) es un procesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos y o aplicaciones 3D interactivas.

HyperThreading Es una marca registrada de la empresa Intel para nombrar su implementación de la tecnología Multithreading Simultáneo también conocido como SMT. Permite a los programas preparados para ejecutar múltiples hilos (multi-threaded) procesarlos en paralelo dentro de un único procesador, incrementando el uso de las unidades de ejecución del procesador.

Malla Poligonal es una colección de triángulos y vértices que aproximan una superficie en 3D.

MultiCore Un microprocesador multinúcleo es aquel que combina dos o más procesadores independientes en un solo paquete, a menudo un solo circuito integrado. Un dispositivo de doble núcleo contiene solamente dos microprocesadores independientes.

NURBS acrónimo inglés de la expresión Non Uniform Rational B-splines, es un modelo matemático muy utilizado en la computación gráfica para generar y representar curvas y superficies.

Octree Una grilla octree es una estructura árbol (informática) de datos en la cual cada nodo interno tiene exactamente 8 hijos. Las grillas octree se usan mayormente para particionar un espacio tridimensional, dividiéndolo recursivamente en ocho octantes. Las grillas octree son las análogas tridimensionales de las grillas quadtree.

OPENGL es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

Pixel Shader sirve para manipular un píxel, o lo que es lo mismo, aplicar un efecto sobre la imagen (realismo, bump mapping, sombras, explosiones y efectos). Se trata de una función gráfica que calcula los efectos sobre una base per-pixel. Dependiendo de la resolución, una cantidad de 2 millones de píxeles puede ser necesario para ser renderizado,iluminado, sombreado, y color para cada marco.

Rasterization es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de píxeles o puntos para ser desplegados en un medio de salida digital, como una pantalla de computadora, una impresora electrónica o una imagen de mapa de bits (bitmap).

SIMD (siglas en inglés de Single Instruction, Multiple Data, en español: Una Instrucción, Múltiples Datos) es una técnica empleada para conseguir paralelismo a nivel de datos.

Vertex Shader es una herramienta capaz de trabajar con la estructura de vértices de figuras modeladas en 3D, y realizar operaciones matemáticas sobre ella para definir colores, texturas e incidencia de la luz. Esto da libertad a los programadores para realizar diferentes efectos, desde la deformación de un objeto hasta la recreación de las olas del mar.

INTRODUCCIÓN

En la computación gráfica uno de los espacios geométricos más utilizado es el euclídeo, no obstante, algunos matemáticos y computistas han encontrado una forma de optimizar y simplificar algunas expresiones usando otros tipos de espacios geométricos. Un ejemplo clásico se puede encontrar en la traslación de un punto, el espacio euclídeo aunque resuelve este problema, no presenta una solución óptima ni elegante para ser implementada en un computador (ya que el cálculo consiste en multiplicar por una matriz y sumar por un vector), por esa razón, se emplea otro tipo de espacio como el proyectivo [1], en este espacio la traslación, rotación y escalado se simplifican en una matriz 4x4, siendo efeciente para ser implementado en un computador.

En ciertas ocasiones, algunos espacios o geometrías no necesariamente optimizan los cálculos, pero si ayudan a expresar ciertas expresiones en forma sencilla o simple, un ejemplo es el modelo conforme. Este modelo es muy atractivo para la computación gráfica, ya que presenta propiedades muy importantes que no todos los espacios posee, como por ejemplo, es homogéneo, preserva distancias y ángulos, entre otras propiedades que serán mencionadas en el capítulo 2. Pero este espacio presenta el inconveniente de trabajar en 5D, haciendo que los cálculos no sean tan rápidos como el espacio euclideo 3D, o el espacio proyectivo 4D.

Al tener este inconveniente, este trabajo implementa los algoritmos usando el GPU (Graphics Processor Unit) para optimizar los tiempos de cómputo en comparación al CPU. En la actualidad la programación por GPU se esta convirtiendo en un estándar. Las empresas diseñadoras de programas, universidades, científicos, entre otros, están utilizando el poder del GPU, por su gran rapidez y simplicidad en su uso.

El cálculo de las intersecciones y detección de colisiones de objetos es ampliamente utilizado en la computación gráfica, tanto en el mundo científico como en el de entretenimiento. En este trabajo se definirán los algoritmos en la geometria conforme para las colisiones de las primitivas básicas como son:

- Intersección Recta-Recta.
- Intersección Recta-Plano.
- Intersección Plano-Plano.
- Intersección Recta-Esfera.
- Intersección Plano-Esfera.
- Intersección Esfera-Esfera.

Estas 6 intersecciones son fundamentales para cualquier librería de algoritmos de colisión, y serán implementadas usando el GPU, comparando los tiempos de cómputo entre el GPU y CPU.

Cabe mencionar que no hay un trabajo previo en el área de colisiones usando el modelo conforme. No obstante, Daniel Fontijne y Leo Dorst [1] emplearon el modelo conforme usando ray tracing. En ese trabajo sólo utilizaron la intersección Recta-Plano y Recta-Esfera.

La estructura de este trabajo se divide de la siguiente manera, los dos primeros capítulos, presentan el marco teórico matemático para resolver las intersecciones, en el primer capítulo se detalla el álgebra geométrica, que es el marco de trabajo que usa la geometría conforme que se estudia en el capítulo 2. El capítulo 3 presenta un marco teórico computacional, explicando el origen de la programación paralela usando el GPU, y los últimos capítulos 4 y 5 detallan la implementación y análisis de los algoritmos creados.

CAPÍTULO I

ÁLGEBRA GEOMÉTRICA

Entre los variados enfoques utlizados en computación gráfica, uno de los más recientes es el uso del Álgebra Geométrica. Se explicará como es utilizada para resolver problemas en el ámbito de la computación gráfica, utilizando sus productos fundamentales, que son el producto externo y el producto geométrico. Cabe mencionar que esta álgebra definida sobre un espacio vectorial $\mathbb{R}^n$, hereda ciertas propiedades de dicho espacio, como lo es la distancia euclidiana, el producto interno, entre otros.

Dado lo extenso de ciertos temas básicos del área, no se tratarán el algebra vectorial, números complejos, quaterniones, geométrica computacional, entre otros. Se asume que el lector posee dichos conocimientos que pueden ser encontrados en las referencias [2], [3], [4], [1], [5].

1 Producto Externo (Outer Product)

Definición 1 *Sean a y b dos vectores, el producto externo viene representado por la siguiente expresión:*

$$a \wedge b \tag{I.1}$$

y se representa con el signo $\wedge$. Esta expresión describe a un plano orientado formado por los vectores a y b, cuya magnitud es el área del paralelogramo que forman dicho vectores.

$$\| a \wedge b \| = \| a \| \| b \| sen\theta \tag{I.2}$$

Es importante recalcar que la expresión $a \wedge b$ no es el producto vectorial en $\mathbb{R}^3$, en primera instancia porque en ningún momento se especifica alguna dimensión para a y b. Y aunque se muestra adelante cierta relación, ambos productos que suelen ser representados por el signo $\wedge$ no son lo mismo. Además el producto vectorial tiene como resultado un vector, mientras que el producto externo es un "plano orientado o hiperplano orientado".

De esto último se tiene que el producto externo no es un vector, ni un escalar. El nombre que se da al producto $a \wedge b$ es ***bivector****.*

Como nota histórica, fue Herman G. Grassmann [1809-1877] en su libro de cálculo geométrico Lineale Ausdehnungslehre quien introduce este producto externo y su significado [6], que luego también es mencionado en su libro Die Ausdehnungslehre, aunque fue el

matemático ingles William Clifford [1845-1879], quién toma las ideas de Grassman y la formaliza en lo que hoy se conoce como álgebra geométrica [7].

Propiedades algebraicas

- $a \wedge b = -b \wedge a$
- $a \wedge (b + c) = a \wedge b + a \wedge c$
- $\|a \wedge a\| = \|a\|\|a\| sin 0 = 0$

La primera propiedad tiene parecido al producto vectorial donde $a \times b = -b \times a$. La segunda da un comportamiento igual al producto escalar y la última indica que el producto externo de dos vectores paralelos es igual 0, ya que dos vectores paralelos geométricamente no forman ningún área.

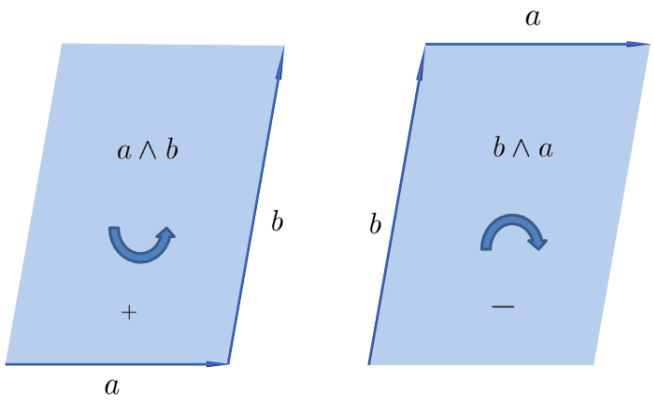


Figura I.1: *Orientación de un bivector*

En la definición anterior se menciona que el plano formado por a y b tiene una orientación. Se conoce que el producto vectorial retorna un vector perpendicular a los dos vectores del producto, la dirección de este vector se crea por la regla de la mano derecha, la cual indica si el vector va en dirección positiva o negativa. Para el producto externo se aplica la misma regla, o lo que es lo mismo se dice que el producto tiene orientación positiva si va en dirección anti-horario o tiene orientación negativa si va en dirección horario. De aquí la expresión $a \wedge b = -b \wedge a$, como se muestra en la figura I.1

Por otra parte, el paralelogramo formado por $a \wedge b$ no es único, en efecto, sea $a' = a + \lambda b$, tenemos:

$$\begin{aligned} a' \wedge b &= (a + \lambda b) \wedge b \\ &= a \wedge b + \lambda b \wedge b \\ &= a \wedge b \end{aligned} \tag{I.3}$$

1.1 Producto externo en 2D y 3D

Una de las ventajas de trabajar en el álgebra geométrica es que funciona en cualquier dimensión, y como ejemplo, se muestra el producto externo para vectores en $\mathbb{R}^2$ y $\mathbb{R}^3$.

Se van utilizar las bases ortonormales de dichos espacios usando la siguiente notación e_1, e_2, e_3.

Sean dos vectores en $\mathbb{R}^2$ escritos de la forma:

$$\begin{aligned} a &= a_1 e_1 + a_2 e_2 \\ b &= b_1 e_1 + b_2 e_2 \end{aligned} \tag{I.4}$$

Aplicando el producto externo se obtiene:

$$\begin{aligned} a \wedge b &= (a_1 e_1 + a_2 e_2) \wedge (b_1 e_1 + b_2 e_2) \\ &= a_1 b_1 (e_1 \wedge e_1) + a_1 b_2 (e_1 \wedge e_2) + a_2 b_1 (e_2 \wedge e_1) + a_2 b_2 (e_2 \wedge e_2) \end{aligned} \tag{I.5}$$

Usando las propiedades algebraicas:

$$e_1 \wedge e_1 = e_2 \wedge e_2 = 0 \quad y \quad e_2 \wedge e_1 = -e_1 \wedge e_2 \tag{I.6}$$

Se tiene que:

$$a \wedge b = (a_1 b_2 - a_2 b_1)(e_1 \wedge e_2) = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix} (e_1 \wedge e_2) \tag{I.7}$$

Esta última expresión guarda relación con los números complejos, ya que el escalar que multiplica al bivector $e_1 \wedge e_2$ es la magnitud del término imaginario que se obtiene al multiplicar un número complejo por el conjugado del segundo número. Por lo tanto, $a \wedge b$ es un área que multiplica a un bivector unitario, que en el caso de $\mathbb{R}^2$ es el plano formado por el bivector $e_1 \wedge e_2$ donde descansan los vectores a y b. A continuación se presenta el caso para $\mathbb{R}^3$.

Sean dos vectores en $\mathbb{R}^3$ escritos de la forma:

$$\begin{aligned} a &= a_1 e_1 + a_2 e_2 + a_3 e_3 \\ b &= b_1 e_1 + b_2 e_2 + b_3 e_3 \end{aligned} \tag{I.8}$$

Aplicando el mismo procedimiento para $\mathbb{R}^2$ y estas propiedades:

$$e_1 \wedge e_1 = e_2 \wedge e_2 = e_3 \wedge e_3 = 0 \tag{I.9}$$

$$e_2 \wedge e_1 = -e_1 \wedge e_2 \quad e_1 \wedge e_3 = -e_3 \wedge e_1 \quad e_3 \wedge e_2 = -e_2 \wedge e_3 \tag{I.10}$$

Se tiene que:

$$a \wedge b = (a_1b_2 - a_2b_1)(e_1 \wedge e_2) + (a_2b_3 - a_3b_2)(e_2 \wedge e_3) + (a_3b_1 - a_1b_3)(e_3 \wedge e_1) \tag{I.11}$$

Los términos que multiplican a cada bivector unitario, es el área del plano proyectado formado por $a \wedge b$ en cada uno de los bivectores, como se muestra en la figura I.2.

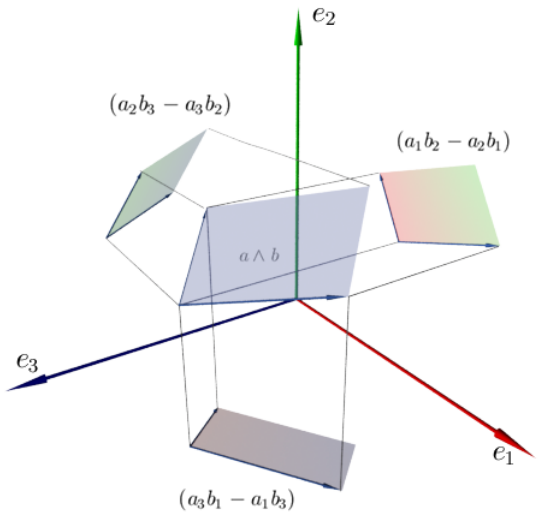


Figura I.2: *Áreas proyectadas de un bivector*

De esto último se obtiene la siguiente igualdad:

$$\|a \wedge b\|^2 = (a_1b_2 - a_2b_1)^2 + (a_2b_3 - a_3b_2)^2 + (a_3b_1 - a_1b_3)^2 \tag{I.12}$$

Es decir, el área al cuadrado del bivector $a \wedge b$ es igual a la suma del cuadrado de todas las áreas proyectadas. La demostración de esta igualdad se puede ver en [3].

2 Producto Geométrico (Geometric Product)

Antes de expresar el producto geométrico, se destaca lo siguiente, sean dos números complejos:

$$\begin{aligned} z_1 &= a_1 + ib_1 \\ z_2 &= a_2 + ib_2 \end{aligned} \tag{I.13}$$

Se tiene que $z_1 z_2^*$ es igual a:

$$z_1 z_2^* = (a_1 a_2 + b_1 b_2) + i(a_2 b_1 - a_1 b_2) \tag{I.14}$$

Esta fórmula indica que $z_1 z_2^*$ es igual a un escalar (producto interno) mas un área (producto externo). William Clifford [3] observando esto, presentó una fórmula que generalizaba dicha ecuación para vectores de cualquier dimensión y que es el producto geométrico.

Definición 2 *Sean dos vectores a y b, su producto geométrico es:*

$$ab = a \cdot b + a \wedge b \tag{I.15}$$

Lo cual es la suma de un escalar y de un bivector. Esta expresión presenta los siguientes axiomas.

- $a(bc) = (ab)c = abc$
- $(\lambda a)b = \lambda(ab) = \lambda ab$
- $a(b + c) = ab + ac$
- $(b + c)a = ba + ca$
- $a^2 = \|a\|^2$

Es decir, este producto es asociativo y distributivo.

Debido a la antisimetría del producto externo se tiene que:

$$\begin{aligned} ba &= b \cdot a + b \wedge a \\ &= a \cdot b - a \wedge b \end{aligned} \tag{I.16}$$

Si ambos vectores son paralelos entonces:

$$a(\lambda a) = \lambda a \cdot a + \lambda a \wedge a = \lambda a \cdot a \tag{I.17}$$

(I.18)

Si los vectores son perpendiculares:

$$ab = a \cdot b + a \wedge b = a \wedge b \tag{I.19}$$

$$ba = b \cdot a + b \wedge a = -a \wedge b = -ab \tag{I.20}$$

Es decir, en el caso de que los vectores sean perpendiculares el producto es anticonmutativo.

Conociendo el producto geométrico, es posible escribir el producto interno y el producto externo en términos de este producto, que viene expresado por las siguientes igualdades:

$$\begin{aligned} a \wedge b &= \frac{1}{2}(ab - ba) \\ a \cdot b &= \frac{1}{2}(ab + ba) \end{aligned} \tag{I.21}$$

2.1 Producto Geométrico de la base

Antes de presentar el producto geométrico de los vectores de la base, se presenta la siguiente notación a utilizar de aquí en adelante.

$$e_i e_j = e_{ij} \tag{I.22}$$

Donde i, j son los índices de la base. A continuación se muestra el producto $e_1 e_1$.

$$e_1 e_1 = e_{11} = e_1 \cdot e_1 + e_1 \wedge e_1 \tag{I.23}$$

Se conoce que $e_1 \cdot e_1 = 1$ y $e_1 \wedge e_1 = 0$. Por lo tanto:

$$e_1 e_1 = e_1^2 = 1 \tag{I.24}$$

Ahora se muestra el producto $e_1 e_2$.

$$e_1 e_2 = e_{12} = e_1 \cdot e_2 + e_1 \wedge e_2 \tag{I.25}$$

Como estos vectores son ortonormales $e_1 \cdot e_2 = 0$, por lo tanto:

$$e_{12} = e_1 \wedge e_2 \tag{I.26}$$

En el caso de tomar el orden inverso se tiene que $e_{21} = -e_{12}$.

2.2 Propiedad imaginaria

Se considera ahora el cálculo del siguiente producto $(e_1 \wedge e_2)^2$.

$$(e_1 \wedge e_2)^2 = (e_1 \wedge e_2)(e_1 \wedge e_2) = e_1 e_2 e_1 e_2 \tag{I.27}$$

Se tiene que

$$e_1 e_2 = -e_2 e_1 \tag{I.28}$$

Por lo tanto

$$(e_1 \wedge e_2)^2 = -e_1 e_1 e_2 e_2 = -e_1^2 e_2^2 = -1 \tag{I.29}$$

Con este resultado se tiene que el álgebra geométrica guarda cierta relación con los números complejos, ya que:

$$(e_1 \wedge e_2) = e_{12} = \sqrt{-1} = I \tag{I.30}$$

En este caso el álgebra geométrica usa la I mayúscula para evitar confusión con la i minúscula usada para representar un número complejo.

De igual manera, si a un vector a se le aplica la siguiente multiplicación aI, el vector de esa multiplicación sufre una rotación de 90° en sentido antihorario, y rotará en sentido horario si se hace la multiplicación Ia. Algo que ocurre también en los números complejos.

2.3 Trivectores

Previamente se menciona que un bivector $a \wedge b$ es un plano orientado cuya norma es el área del paralelogramo que forman dichos vectores. Es natural pensar cual sería el significado de esta expresión $a \wedge b \wedge c$ cuyo nombre es trivector.

Definición 3 *Un trivector representado de esta forma:*

$$a \wedge b \wedge c \tag{I.31}$$

Es un bivector $a \wedge b$ que representa un área, que se mueve a largo de un tercer vector c, formando así un paralelepípedo y por lo tanto un volumen.

Se tienen distintas maneras de agrupar los vectores para producir el mismo paralelepípedo, por ejemplo, es claro que:

$$(a \wedge b) \wedge c = (b \wedge c) \wedge a = (c \wedge a) \wedge b \tag{I.32}$$

Como se muestra en la figura I.3:

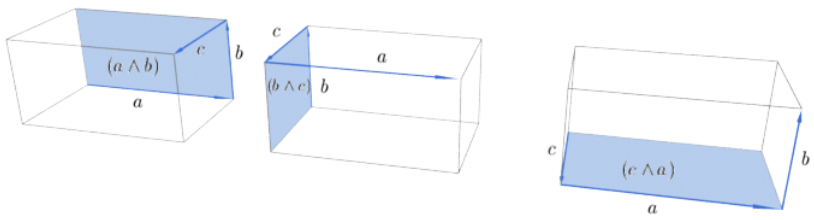


Figura I.3: *Volumen de un trivector*

Consideremos los vectores a, b y c escritos como combinación lineal de las bases:

$$\begin{aligned} a &= a_1e_1 + a_2e_2 + a_3e_3 \\ b &= b_1e_1 + b_2e_2 + b_3e_3 \\ c &= c_1e_1 + c_2e_2 + c_3e_3 \end{aligned} \tag{I.33}$$

Al resolver $a \wedge b \wedge c$ se obtiene expresiones como:

$$e_1 \wedge e_1 \wedge e_1 = 0 \quad e_2 \wedge e_2 \wedge e_2 = 0 \quad e_3 \wedge e_3 \wedge e_3 = 0 \tag{I.34}$$

Que claramente son 0, pero a su vez se tiene nuevas expresiones como:

$$e_1 \wedge e_2 \wedge e_3 \quad e_1 \wedge e_2 \wedge e_1 \quad e_1 \wedge e_2 \wedge e_2, \quad etc. \tag{I.35}$$

Algunas de estas expresiones son igual a 0, por ejemplo, la segunda expresión se puede leer como el bivector $e_1 \wedge e_2$ que se desplaza a través del vector e_1 lo cual claramente es 0.

Aplicando lo ya mencionado y lo visto para el caso de los bivectores resulta que:

$$a \wedge b \wedge c = \begin{vmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{vmatrix} (e_1 \wedge e_2 \wedge e_3) \tag{I.36}$$

Este determinante en la expresión de arriba claramente es el volumen del paralelepípedo que se forman con los vectores a, b y c. Cabe destacar que en $\mathbb{R}^3$ también se tiene propiedades imaginarias, es decir:

$$\begin{aligned} (e_1 \wedge e_2)^2 &= -1 \\ (e_2 \wedge e_3)^2 &= -1 \\ (e_3 \wedge e_3)^2 &= -1 \end{aligned} \tag{I.37}$$

Mas aún,

$$(e_1 \wedge e_2 \wedge e_3)^2 \ = -1 \tag{I.38}$$

Definición 4 *El álgebra geométrica usa el término* ***grado*** *para distinguir sus elementos, por ejemplo, un escalar es de grado-0, un vector de grado-1, un bivector de grado-2, un trivector de grado-3, y así sucesivamente. En cada álgebra se define pseudoescalar como el elemento de mayor grado del álgebra.*

Por ejemplo, en $\mathbb{R}^2$ el álgebra posee los siguientes elementos:

Elemento	Símbolo	Grado
1 escalar	λ	0
2 vectores	e_1, e_2	1
1 bivector unit.	$e_1 \wedge e_2 = e_{12}$	2

Y el pseudoescalar en $\mathbb{R}^2$ sería el bivector unitario $e_1 \wedge e_2 = e_{12}$. En el caso de $\mathbb{R}^3$ los elementos son:

Elemento	Símbolo	Grado
1 escalar	λ	0
3 vectores	e_1, e_2, e_3	1
3 bivectores	$e_1 \wedge e_2 = e_{12}$	2
	$e_2 \wedge e_3 = e_{23}$	
	$e_3 \wedge e_1 = e_{31}$	
1 trivector	$e_1 \wedge e_2 \wedge e_3 = e_{123}$	3

Y el pseudoescalar es el trivector e_{123}.

Definición 5 *Un multivector es una combinación lineal de los elementos del álgebra. Por ejemplo, en $\mathbb{R}^2$ un multivector se escribe como:*

$$A = \lambda_0 + \lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_{12} \tag{I.39}$$

Estos multivectores, se pueden sumar, restar y multiplicar.

Definición 6 *El álgebra geométrica permite la división de vectores y bivectores, por lo tanto la inversa de un vector a viene expresada de la siguiente manera:*

$$a^{-1} = \frac{a}{a^2} = \frac{a}{\|a\|^2} \tag{I.40}$$

La misma expresión se puede utilizar para calcular la inversa de un bivector B.

$$B^{-1} = \frac{B}{B^2} = \frac{B}{\|B\|^2} \tag{I.41}$$

3 Dualidad

A continuación se muestran los siguientes productos, considerando en primera instancia $\mathbb{R}^2$ y $I = e_1 \wedge e_2$.

$$\begin{aligned} Ie_1 &= e_1e_2e_1 = -e_2 \\ Ie_2 &= e_1e_2e_2 = e_1 \\ \\ e_1I &= e_1e_1e_2 = e_2 \\ e_2I &= e_2e_1e_2 = -e_1 \end{aligned} \tag{I.42}$$

En este ejemplo, se observa que $Ie_i = -e_iI$. Ahora se muestra para el caso de $\mathbb{R}^3$ y $I = e_1 \wedge e_2 \wedge e_3$.

$$\begin{aligned} Ie_1 &= e_1e_2e_3e_1 = e_2e_3 \\ Ie_2 &= e_1e_2e_3e_2 = e_3e_1 \\ Ie_3 &= e_1e_2e_3e_3 = e_1e_2 \\ \\ e_1I &= e_1e_1e_2e_3 = e_2e_3 \\ e_2I &= e_2e_1e_2e_3 = e_3e_1 \\ e_3I &= e_3e_1e_2e_3 = e_1e_2 \end{aligned} \tag{I.43}$$

En el caso de $\mathbb{R}^3$, $Ie_i = e_iI$, es decir, cuando el espacio sea de dimensión par, la multiplicación es anticonmutativa $Ia = -aI$, y en el caso de que sea impar la multiplicación es conmutativa $Ia = aI$. La generalización para cualquier espacio de dimensión n viene dado por la expresión:

$$I_nA^r = (-1)^{r(n-1)}A_rI_n \tag{I.44}$$

Otro punto importante a destacar es el siguiente, el resultado de estás operaciones es perpendicular al valor que multiplica el pseudoescalar, por ejemplo, $Ie_1 = e_1e_2e_3e_1 = e_2e_3$ donde el bivector resultante es perpendicular a e_1.

Definición 7 *La operación de multiplicar un elemento del álgebra por el pseudoescalar se define como transformación dual.*

$$aI = a \cdot I \tag{I.45}$$

Esta operación sera conmutativa en espacio de dimensiones impares, y anticonmutativa en espacio de dimensión par.

4 Reflexión y Rotaciones

Existen varias maneras de rotar un objeto en 3D, ya sea al utilizar una notación matricial [8], o se puede usar el poder de los quaterniones [9]. Este último es uno de los más utilizados en el ámbito de la computación gráfica y video juegos. A continuación se presenta como el álgebra geométrica resuelve estos problemas, no se pretende entrar en detalle en las demostraciones sino mostrar solo el resultado que nos permita realizar estas operaciones.

4.1 Reflexión

Sea la figura I.4, la siguiente fórmula es utilizada para reflejar un vector a, en un plano de vector normal n, la notación $\hat{n}$ representa el vector normal unitario.

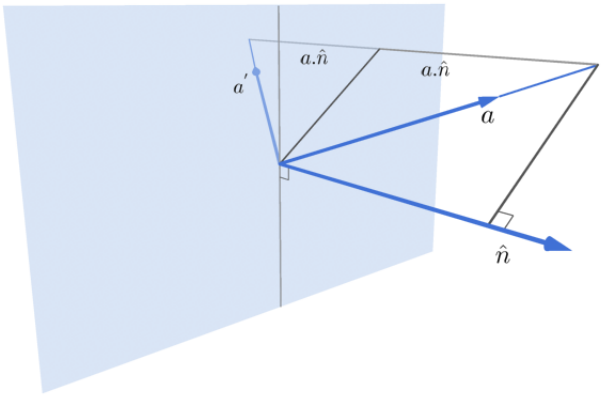


Figura I.4: *Reflexión de un vector*

$$a^{'} = a - (2a.\hat{n})\hat{n} \tag{I.46}$$

Posteriormente usando el álgebra geométrica sería:

$$a^{'} = -\hat{n}a\hat{n} \tag{I.47}$$

Esta fórmula tiene una estructura muy parecida a la expresión para rotar un vector usando quaterniones. En el caso de que se quiera reflejar un bivector $B = a \wedge b$, se tendría que aplicar la fórmula I.47 a cada vector del bivector.

$$\begin{aligned} a^{'} &= -\hat{n}a\hat{n} \\ b^{'} &= -\hat{n}b\hat{n} \end{aligned} \tag{I.48}$$

Por lo tanto

$$\begin{aligned} B^{'} &= (-\hat{n}a\hat{n}) \wedge (-\hat{n}b\hat{n}) \\ B^{'} &= (\hat{n}a\hat{n}) \wedge (\hat{n}b\hat{n}) \end{aligned} \tag{I.49}$$

O lo que es lo mismo

$$B^{'} = \hat{n}B\hat{n} \tag{I.50}$$

La demostración de estas fórmulas pueden encontrarse en la referencia [3].

4.2 Rotación

Una rotación puede ser vista como la ejecución de dos reflexiones. Sea la figura I.5, donde tenemos dos rectas n y m, creado a partir de dos vectores unitarios $\hat{n}$ y $\hat{m}$ respectivamente. Y el ángulo de dicho vectores es θ.

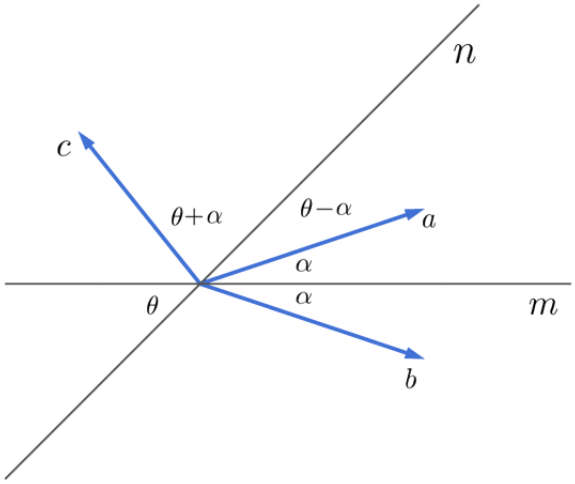


Figura I.5: *Rotación de un vector*

El vector a es el que se quiere rotar, primero se refleja en m formando así el vector b, y luego este vector se refleja en n, creando el vector c, este último vector forma un ángulo con respecto a a de 2θ. En términos del álgebra geométrica se tiene que:

$$c = \hat{n}\hat{m}a\hat{m}\hat{n} \tag{I.51}$$

Si establece $R = \hat{n}\hat{m}$

$$c = Ra\tilde{R} \tag{I.52}$$

Donde $\tilde{R} = \hat{m}\hat{n}$ se define como reversión de R y a R se le llama rotor. Cabe mencionar que esto funciona para cualquier dimensión, además tiene cierto parecido a los números complejos al poder expresar R en términos de la exponencial.

$$c = e^{-\hat{B}\frac{\theta}{2}} a e^{\hat{B}\frac{\theta}{2}} \tag{I.53}$$

Donde $\hat{B}$ es el bivector unitario formado por $\hat{n}$ y $\hat{m}$. Además se tiene que:

$$e^{-\hat{B}\frac{\theta}{2}} = cos\left(\frac{\theta}{2}\right) - \hat{B}sin\left(\frac{\theta}{2}\right) \tag{I.54}$$

La demostración de estas expresiones, pueden encontrarse en la referencia [3].

5 Algunas aplicaciones del álgebra geométrica

A continuación se presentan algunas aplicaciones del álgebra geométrica para resolver ciertos cálculos en la computación gráfica. En esta sección se presenta sólo algunos ejemplos, ya que son muchos los problemas los que se pueden resolver usando esta álgebra.

5.1 Punto dentro de un triángulo

Sea el triángulo de la figura I.6. Para saber si el punto P_0 puede estar en el triángulo formado por P_1, P_2, P_3, primero se calcula el volumen del trivector.

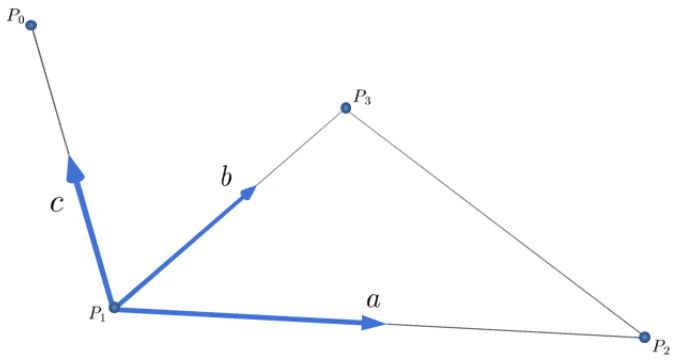


Figura I.6: *Punto fuera de un triángulo*

$$volumen = a \wedge b \wedge c$$

Como se expresó en la ecuación I.32 si este volumen es 0 implica que el punto esta en el plano, más no asegura que este dentro del triángulo. Para ello se muestra la siguiente figura, suponiendo que el volumen es 0.

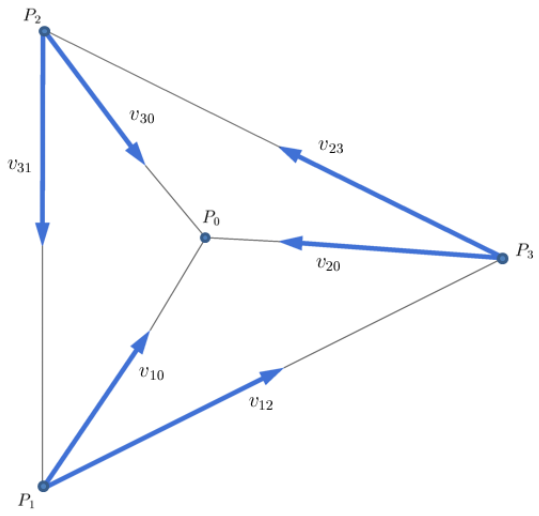


Figura I.7: *Punto dentro de un triángulo*

El paso siguiente es resolver las siguientes expresiones:

$$v_{12} \wedge v_{10} \quad v_{23} \wedge v_{20} \quad v_{31} \wedge v_{30}$$

Cada uno de estos bivectores contiene una combinación lineal de los bivectores de la base, como se vio en la expresión I.11. Si el signo de los escalares que acompañan a estos bivectores son todos positivos en los 3 productos externos, entonces el punto se encuentra dentro del triángulo. Si en uno de estos productos, el signo es negativo, entonces el punto reside afuera del triángulo, y en el caso que una de las áreas sea igual a 0, implica que el punto yace en uno de los bordes.

5.2 Orientación de un punto con relación a una recta

La figura I.8 muestra el siguiente escenario, T y P son puntos de la recta, t, p sus respectivos vectores y $v = p - t$.

Es claro, la siguiente expresión:

$$v \wedge (p - t) = 0 \tag{I.55}$$

Ahora se sustituye p por un punto Q cualquiera, se tiene que el punto Q está en la recta si:

$$v \wedge (q - t) = 0 \tag{I.56}$$

Si Q se ubica por encima de la recta, la expresión anterior da positivo, y en el caso de ser negativo estaría por debajo de la recta. La decisión de si está por debajo o por

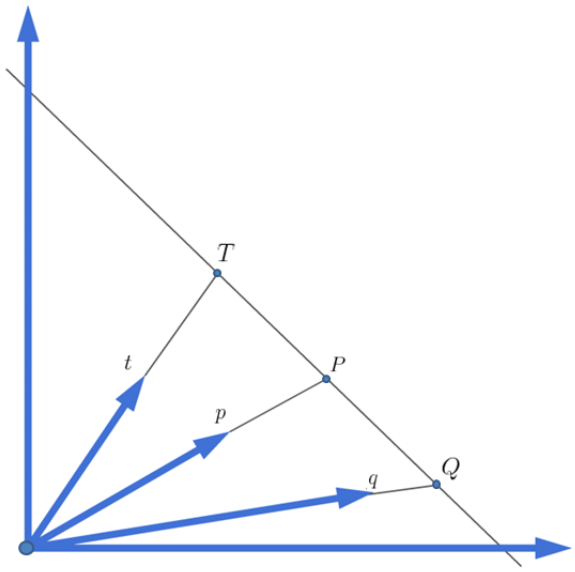


Figura I.8: *Orientación de un punto con respecto a una recta*

arriba, depende de la orientación horario o antihorario, como se explicó anteriormente en la sección de bivectores.

5.3 Orientación de un punto con relación a un plano

En el caso de un plano es muy similar al anterior, sea la figura I.9, donde se tiene un plano formado por un bivector A, y los punto T y P yacen en el plano.

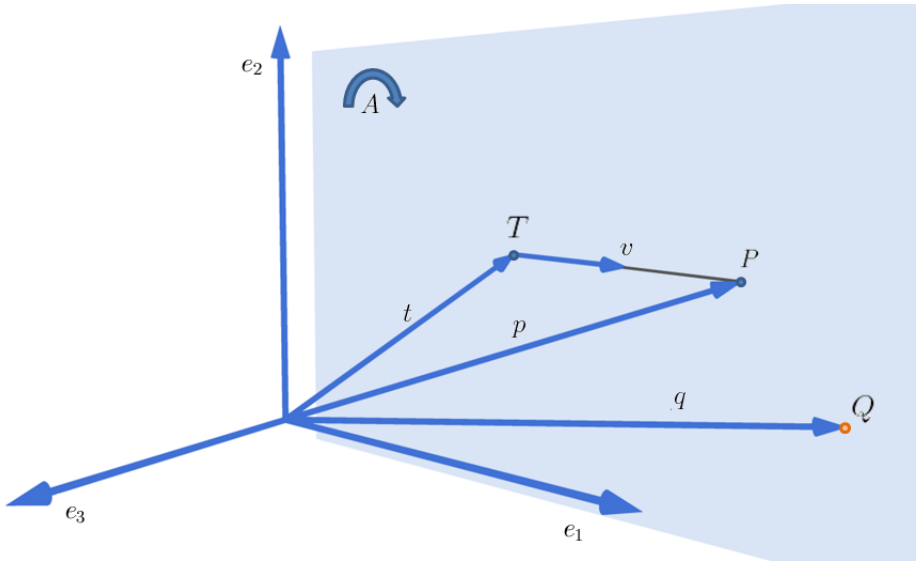


Figura I.9: *Orientación de un punto con respecto a un plano A*

La siguiente expresión es clara, por ser $p - t$ paralelo al bivector A.

$$A \wedge (p - t) = 0 \tag{I.57}$$

Por lo tanto sea un punto Q cualquiera bastaría resolver esta expresión

$$A \wedge (q - t) = 0 \tag{I.58}$$

Y se aplica el mismo criterio empleado con la recta, para determinar si el punto yace en el plano, o se encuentra en algunos de los lados del mismo.

5.4 Distancia más corta de un punto a un plano

Sea el siguiente escenario mostrado en la figura I.10, se quiere encontrar la distancia más corta entre el punto P y el plano formado por A, o lo que es lo mismo, se quiere encontrar $\|d\|$.

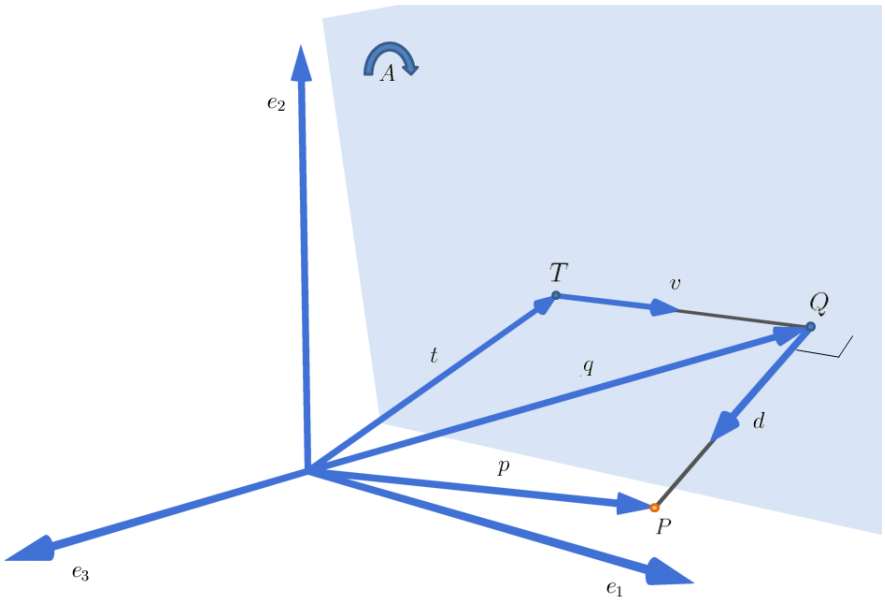


Figura I.10: *Corta distancia de un punto a un plano*

De la figura se deduce la siguiente expresión:

$$d = p - t - v \tag{I.59}$$

Ahora

$$A \wedge v = 0 \tag{I.60}$$

Como el objetivo es encontrar la norma de d, se utiliza la siguiente expresión.

$$Ad = A \cdot d + A \wedge d \tag{I.61}$$

Pero d es perpendicular al plano A, por lo tanto el producto interno es 0.

$$Ad = A \wedge d \tag{I.62}$$

Pero en esta álgebra se tiene inversa, por lo tanto,

$$A^{-1}Ad = A^{-1}(A \wedge d) \tag{I.63}$$

Lo que implica

$$d = A^{-1}(A \wedge d) \tag{I.64}$$

Sustituyendo el valor de d,

$$d = A^{-1}(A \wedge (p - t - v)) \tag{I.65}$$

Resolviendo queda,

$$d = A^{-1}(A \wedge (p - t)) \tag{I.66}$$

Y así se obtiene d y por consiguiente se puede calcular su norma.

CAPÍTULO II

GEOMETRÍA CONFORME

En el área de la computación gráfica (CG) es muy común el uso del espacio euclidiano para resolver la mayoría de los problemas, no obstante, algunos problemas pueden simplificarse o hacerse más eficiente en otros espacios, como por ejemplo, en el espacio proyectivo. Un ejemplo clásico en la computación gráfica es la traslación, dicha transformación en el espacio euclideo es no lineal haciendo que no se puede expresar la traslación, rotación y escalado en una sola matriz, pero es en el espacio proyectivo [10], con el uso de coordenadas homogéneas que se puede expresar estas 3 transformaciones en una sola matriz 4x4 simplificando así los cálculos.

Mas aún, la librería OpenGL muy utilizada en el desarrollo de programas gráficos trabaja con el modelo proyectivo. Pero este modelo presenta algunas desventajas, como por ejemplo, no se pueden representar círculos y esferas en dicho espacio, y la distancia del espacio euclideo no se preserva en el espacio proyectivo.

La geométria conforme se muestra como una alternativa que se ha venido estudiando muy recientemente en el área de la computación gráfica [11]. Este modelo, usando el álgebra geométrica como marco de trabajo, se presenta como una alternativa del espacio euclidiano para resolver problemas de computación gráficas en 3D y presenta características que lo hacen muy atractivo para resolver problemas matemáticos, siendo algunos de ellos:

- Homogeneidad.
- Soporta puntos y líneas en el infinito.
- Provee de un mecanismo muy simple para representar objetos geométricos, como esferas, círculos, lineas, etc.
- Preserva ángulos y distancias.

Aunque estas propiedades son muy atractivas, realizar cálculos en dicho espacio puede ser una tarea ardua, como se muestra en este capítulo. Pero esto no quita la elegancia en la manera en como esta geometría trabaja los problemas.

Uno de los problemas a resolver son las siguientes seis intersecciones:

- Recta-Recta.
- Recta-Plano.

- Plano-Plano.
- Recta-Esfera.
- Plano-Esfera.
- Esfera-Esfera.

Estas seis intersecciones son vitales en la computación gráfica, y muy utilizadas en otros ámbitos, como vídeo juegos, producciones cinematográficas, investigaciones científicas, etc. El propósito de este capítulo es resolver estas seis intersecciones, y presentar también otros cálculos necesarios en la CG como lo son la traslación, rotación, escalado y reflexión.

No se pretende entrar a fondo en como el modelo conforme se crea o se genera, pero para los interesados, una excelente explicación de esto puede ser encontrado en los libros de Chris Doran y Anthony Lasenby [12] y L. Dorst, D. Fontijne y S. Mann [13].

1 Modelo Conforme

Usualmente en el espacio Euclideo 3D se trabaja con una base ortonormal, es decir, $e_1.e_1 = e_2.e_2 = e_3.e_3 = 1$, pero no necesariamente en un espacio el producto interno de las bases tiene que ser positivo. Fue Herman Minkowski, quién presentó un espacio de 4D, tal que $e_4.e_4 = -1$, estos valores positivos y negativos definen el signo de un espacio y se escribe de la siguiente manera $\mathbb{R}^{p,q}$, donde p son las dimensiones positivas y q las dimensiones negativas.

Definición 8 *Un vector nulo X, es aquel cuya norma es igual a 0.*

$$X \cdot X = \|X\|^2 = 0 \tag{II.1}$$

Donde no necesariamente $X = 0$. Estos vectores pueden existir en espacios que tienen signos mixtos.

Definición 9 *En la geometría conforme puntos en el espacio $\mathbb{R}^{p,q}$, son representados como vectores nulos del espacio $\mathbb{R}^{p+1,q+1}$. Por lo tanto, puntos del espacio euclideo $\mathbb{R}^{3,0}$ son representados como vectores nulos del espacio conforme $\mathbb{R}^{4,1}$. Esto da una geometría de 5-dimensiones.*

La base de esta geometría conforme se representa por el siguiente conjunto de vectores $(e_1, e_2, e_3, e, \bar{e})$ ortonormales. Donde:

$$e_1 \cdot e_1 = e_2 \cdot e_2 = e_3 \cdot e_3 = e \cdot e = 1 \;\; y \;\; \bar{e} \cdot \bar{e} = -1 \tag{II.2}$$

Definición 10 *La geometria conforme define los siguientes vectores nulos n y $\bar{n}$ de la siguiente manera:*

$$n = e + \bar{e} \tag{II.3}$$
$$\bar{n} = e - \bar{e} \tag{II.4}$$

Donde n es el vector en el infinito y $\bar{n}$ es el vector origen. Mas adelante se explica el porque se le da estos nombres.

Usando lo visto en el capítulo I, se definen las siguientes tablas (1.1):

GP	e	e_i	$\bar{e}$
e	1	$-e_i e$	$e\bar{e}$
e_i	$e_i e$	1	$e_i\bar{e}$
$\bar{e}$	$-e\bar{e}$	$-e_i\bar{e}$	-1

$.$	e	e_i	$\bar{e}$
e	1	0	0
e_i	0	1	0
$\bar{e}$	0	0	-1

$\wedge$	e	e_i	$\bar{e}$
e	0	$-e_i \wedge e$	$e \wedge \bar{e}$
e_i	$e_i \wedge e$	0	$e_i \wedge \bar{e}$
$\bar{e}$	$-e \wedge \bar{e}$	$-e_i \wedge \bar{e}$	0

GP	n	e_i	$\bar{n}$
n	0	$-e_i n$	$2 - 2e\bar{e}$
e_i	$e_i n$	1	$e_i\bar{n}$
$\bar{n}$	$2 - 2e\bar{e}$	$-e_i\bar{n}$	0

$.$	n	e_i	$\bar{n}$
n	0	0	2
e_i	0	1	0
$\bar{n}$	2	0	0

$\wedge$	n	e_i	$\bar{n}$
n	0	$-e_i \wedge n$	$-2e\bar{e}$
e_i	$e_i \wedge n$	0	$e_i \wedge \bar{n}$
$\bar{n}$	$2e\bar{e}$	$-e_i \wedge \bar{n}$	0

Algunos de estos resultados se encuentran demostrado en la referencia [3], y además serán utilizados para calcular las intersecciones antes mencionadas.

1.1 Elementos del modelo conforme

La definición 2, muestra que este modelo posee 5 dimensiones, esto implica que un multivector en esta geometría viene representado por 32 elementos: 1 escalar, 5 vectores, 10 bivectores, 10 trivectores, 5 quadvectores y 1 pseudoescalar. Que se muestran en la siguiente tabla:

Elemento	Símbolo
1 escalar	λ
5 vectores	$e_1,\ e_2,\ e_3,\ e,\ \bar{e}$
10 bivectores	$e_1 \wedge e_2,\ e_2 \wedge e_3,\ e_3 \wedge e_1,\ e_1 \wedge \bar{e},\ e_2 \wedge \bar{e},\ e_3 \wedge \bar{e},\ e_1 \wedge e,\ e_2 \wedge e$ $e_3 \wedge e,\ \bar{e} \wedge e$
10 trivectores	$e_1 \wedge e_2 \wedge e_3,\ e_1 \wedge e_2 \wedge \bar{e},\ e_1 \wedge e_2 \wedge e,\ e_1 \wedge e_3 \wedge \bar{e},\ e_1 \wedge e_3 \wedge e$ $e_2 \wedge e_3 \wedge \bar{e},\ e_2 \wedge e_3 \wedge e,\ e_1 \wedge \bar{e} \wedge e,\ e_2 \wedge \bar{e} \wedge e,\ e_3 \wedge \bar{e} \wedge e$
5 quadvectores	$e_1 \wedge e_2 \wedge e_3 \wedge \bar{e},\ e_1 \wedge e_2 \wedge e_3 \wedge e,\ e_1 \wedge e_2 \wedge \bar{e} \wedge e,\ e_1 \wedge e_3 \wedge \bar{e} \wedge e$ $e_2 \wedge e_3 \wedge \bar{e} \wedge e$
1 pseudoescalar	$e_1 \wedge e_2 \wedge e_3 \wedge \bar{e} \wedge e$

Una vez definido los elementos de este espacio, se procede a describir las primitivas que residen en este espacio y su funcionamiento.

1.2 Punto

Un punto en el espacio conforme se define como el vector nulo:

$$F(x) = X = 2x + x^2 n - \bar{n} \tag{II.5}$$

Donde x en este caso es un punto en $\mathbb{R}^3$.

$$x = x_1 e_1 + x_2 e_2 + x_3 e_3 \tag{II.6}$$

Ejemplo, sea un punto $x = (1, 0, 2)$ en $\mathbb{R}^3$, su vector nulo correspondiente en el espacio conforme:

$$\begin{aligned} X &= 2(e_1 + 2e_3) + 5n - \bar{n} \\ &= 2e_1 + 4e_3 + 5n - \bar{n} \end{aligned} \tag{II.7}$$

En la definición 3 se señala que n representa el infinito y $\bar{n}$ el origen, a continuación se muestra el por qué de esta asignación. Si $x = (0, 0, 0)$, se tiene:

$$X = -\bar{n} \tag{II.8}$$

De aquí que $\bar{n}$ apunte al origen. Ahora sea la siguiente expresión:

$$X \cdot \bar{n} = 2(x \cdot \bar{n}) + x^2(\bar{n} \cdot n) - \bar{n} \cdot \bar{n} = 2x^2 \tag{II.9}$$

Por lo tanto:

$$\begin{aligned} 2\frac{X}{X \cdot \bar{n}} &= 2\frac{2x + x^2 n - \bar{n}}{2x^2} \\ &= n + \frac{2x - \bar{n}}{x^2} \end{aligned} \tag{II.10}$$

Es claro, que si $x \to \infty$ la expresión $\frac{2x - \bar{n}}{x^2} \to 0$. De aquí que n apunta hacia el infinito.

Sea X e Y dos puntos del espacio conforme, se tiene las siguientes expresiones [3]:

$$X \cdot Y = -2(x - y)^2 \tag{II.11}$$

$$\begin{aligned} X \wedge Y =& 4(x \wedge y) + 2y^2(x \wedge n) - 2(x \wedge \bar{n}) - 2x^2(y \wedge n) \\ & - x^2(n \wedge \bar{n}) + 2(y \wedge \bar{n}) + y^2(n \wedge \bar{n}) \end{aligned} \tag{II.12}$$

Destacando que la ecuación (II.12) señala que el producto interno de X e Y guarda relación con la distancia de ambos puntos en $\mathbb{R}^3$.

1.3 Recta

Sean dos puntos en el espacio conforme P_1 y P_2:

$$P_1 = 2x_1 + x_1^2 n - \bar{n} \tag{II.13}$$

$$P_2 = 2x_2 + x_2^2 n - \bar{n} \tag{II.14}$$

Una recta en el modelo conforme se representa por la siguiente expresión:

$$L = P_1 \wedge P_2 \wedge n \tag{II.15}$$

$$L = 4(x_1 \wedge x_2 \wedge n) - 2(x_2 - x_1) \wedge n \wedge \bar{n} \tag{II.16}$$

La demostración de esta expresión se encuentra en el apéndice A.

Si la recta en $\mathbb{R}^3$, pasa por el origen, es decir, $x_1 = \alpha x_2$. El primer trivector se hace 0, por lo tanto, L para una recta por el origen en $\mathbb{R}^3$, se expresa:

$$\begin{aligned} L &= -2(x_2 - x_1) \wedge n \wedge \bar{n} \\ &= -2(\alpha x_1 - x_1) \wedge n \wedge \bar{n} \\ &= -2(\alpha - 1) x_1 \wedge n \wedge \bar{n} \end{aligned} \tag{II.17}$$

Asimismo, en la ecuación II.16 se puede notar que el segundo trivector guarda relación con el vector dirección de la recta.

Se muestra el siguiente ejemplo, sean dos punto en $\mathbb{R}^3$, $x_1 = (1, 1, 0)$ y $x_2 = (1, 0, 0)$, se tiene que la recta viene representada por:

$$\begin{aligned} L &= 4(e_1 + e_2) \wedge e_1 \wedge n - 2(e_1 - e_1 - e_2) \wedge n \wedge \bar{n} \\ &= 4e_2 \wedge e_1 \wedge n + 2e_2 \wedge n \wedge \bar{n} \end{aligned} \tag{II.18}$$

Un último punto a considerar es el siguiente, sea un tercer punto en el espacio conforme X tenemos que si $L \wedge X = 0$, entonces X vive en la recta. En el caso que $L \wedge X \neq 0$ implica que X no vive en la recta.

Siguiendo el ejemplo de arriba, se considera el punto $x = (1, -1, 0)$ que claramente vive en la recta, el punto conforme asociado es $X = 2(e_1 - e_2) + 2n - \bar{n}$. Se calcula $L \wedge X$:

$$\begin{aligned} L &= (4e_2 \wedge e_1 \wedge n + 2e_2 \wedge n \wedge \bar{n}) \wedge (2(e_1 - e_2) + 2n - \bar{n}) \\ &= 8(e_2 \wedge e_1 \wedge n) \wedge (e_1 \wedge e_2) + 8e_2 \wedge e_1 \wedge n \wedge n - 4e_2 \wedge e_1 \wedge n \wedge \bar{n} + \\ &\quad + 4(e_2 \wedge n \wedge \bar{n}) \wedge (e_1 \wedge e_2) - 4e_2 \wedge n \wedge \bar{n} \wedge n + 2e_2 \wedge n \wedge \bar{n} \wedge \bar{n} \\ &= -4e_2 \wedge e_1 \wedge n \wedge \bar{n} + 4e_2 \wedge n \wedge \bar{n} \wedge e_1 \\ &= -4e_2 \wedge e_1 \wedge n \wedge \bar{n} + 4e_2 \wedge e_1 \wedge n \wedge \bar{n} \\ &= 0 \end{aligned} \tag{II.19}$$

Lo cual verifica que el punto reside en la recta. Ahora se considera el punto $x = (0, 2, 0)$ que no vive en la recta, su punto conforme es $X = 2e_2 + 4n - \bar{n}$. Por lo tanto $L \wedge X$:

$$\begin{aligned} L &= (4e_2 \wedge e_1 \wedge n + 2e_2 \wedge n \wedge \bar{n}) \wedge (2(e_2) + 2n - \bar{n}) \\ &= -4(e_2 \wedge e_1 \wedge n \wedge \bar{n}) \\ &\neq 0 \end{aligned} \tag{II.20}$$

Lo que verifica lo antes mencionado.

1.4 Círculo

Aunque no se va a trabajar con esta primitiva, ya que su uso en la computación gráfica para las operaciones básicas no es común, igual se define como un trivector:

$$C = P_1 \wedge P_2 \wedge P_3 \tag{II.21}$$

Donde P_1, P_2 y P_3 son puntos en el modelo conforme.

El radio se obtiene con la siguiente fórmula:

$$\rho^2 = \frac{-C^2}{(C \wedge n)^2} \tag{II.22}$$

Y su centro es:

$$\varepsilon = CnC \tag{II.23}$$

1.5 Plano

Sean tres puntos en el espacio conforme P_1, P_2 y P_3. Un plano en el modelo conforme se define con la siguiente expresión:

$$\pi = P_1 \wedge P_2 \wedge P_3 \wedge n \tag{II.24}$$

Esta expresión es simple para su uso en computación gráfica, ya que la mayoría de los cálculos con mallas geométricas se realizan con triángulos.

Resolviendo L, si tiene que:

$$\begin{aligned}\pi = &8(x_1 \wedge x_2 \wedge x_3 \wedge n) \\ &+ 4\left[(x_1 \wedge x_2 \wedge n \wedge \bar{n}) - (x_1 \wedge x_2 \wedge n \wedge \bar{n}) + (x_2 \wedge x_3 \wedge n \wedge \bar{n})\right]\end{aligned} \tag{II.25}$$

Esta demostración se encuentra en el apéndice A. Cabe mencionar que para verificar si un punto se encuentra en el plano se verifica que $L \wedge X = 0$, donde X en un vector nulo en el espacio conforme, en caso de que no sea igual a 0 entonces el punto reside fuera del plano.

1.6 Esfera

Una esfera se define en el modelo conforme como un quadvector:

$$S = P_1 \wedge P_2 \wedge P_3 \wedge P_4 \tag{II.26}$$

Donde P_1, P_2, P_3 y P_4 son puntos en el modelo conforme.

Resolviendo S, se tiene que:

$$\begin{aligned}S = &16(x_1 \wedge x_2 \wedge x_3 \wedge x_4) - 8x_3^2(x_1 \wedge x_2 \wedge x_4 \wedge n) + 8(x_1 \wedge x_2 \wedge x_4 \wedge \bar{n}) \\ &- 8x_1^2(x_2 \wedge x_3 \wedge x_4 \wedge n) + 8(x_2 \wedge x_3 \wedge x_4 \wedge \bar{n}) - 8x_2^2(x_3 \wedge x_1 \wedge x_4 \wedge n) \\ &+ 8(x_3 \wedge x_1 \wedge x_4 \wedge \bar{n}) + (4x_3^2 - 4x_2^2)(x_1 \wedge x_4 \wedge n \wedge \bar{n}) + (4x_1^2 - 4x_3^2)(x_2 \wedge x_4 \wedge n \wedge \bar{n}) \\ &(4x_2^2 - 4x_1^2)(x_3 \wedge x_4 \wedge n \wedge \bar{n}) + 8x_4^2(x_1 \wedge x_2 \wedge x_3 \wedge n) + (4x_4^2 - 4x_3^2)(x_1 \wedge x_2 \wedge n \wedge \bar{n}) \\ &(4x_4^2 - 4x_1^2)(x_2 \wedge x_3 \wedge n \wedge \bar{n}) + (4x_4^2 - 4x_2^2)(x_3 \wedge x_1 \wedge n \wedge \bar{n}) - 8(x_1 \wedge x_2 \wedge x_3 \wedge \bar{n})\end{aligned}$$

Esta demostración se encuentra en el apéndice A.

El radio de la esfera viene dada por la siguiente expresión [3]:

$$\rho^2 = \frac{-S^2}{(S \wedge n)^2} \tag{II.27}$$

Y el centro es:

$$\varepsilon = SnS \tag{II.28}$$

2 Transformaciones

2.1 Traslación

En el capítulo 1, se mencionó que un rotor es de la forma:

$$R = e^{-\hat{B}\frac{\theta}{2}} \tag{II.29}$$

Con este rotor se rota un vector en $\mathbb{R}^3$ en un angulo θ. En el modelo conforme una traslación toma la misma noción de los rotores. Sea R definido de la siguiente manera:

$$R = e^{\frac{na}{2}} \tag{II.30}$$

Donde n es el vector infinito y a es un vector en el espacio $\mathbb{R}^{3,0}$. Es claro que a y n son ortogonales, por lo tanto su producto interno $a \cdot n = 0$. Entonces:

$$(na)^2 = nana = -anna = 0 \tag{II.31}$$

Si la ecuación II.30 se reescribe utilizando la serie de Taylor, tenemos:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + ... \tag{II.32}$$

Pero a partir del segundo término, todos se anulan.

$$T_a = e^{\frac{na}{2}} = 1 + \frac{na}{2} \tag{II.33}$$

$$\tilde{T_a} = 1 - \frac{na}{2} \tag{II.34}$$

Ahora se muestran las siguientes expresiones:

$$\begin{aligned} T_a x \tilde{T_a} &= x + n(a \cdot x) \\ T_a n \tilde{T_a} &= n \\ T_a \bar{n} \tilde{T_a} &= \bar{n} - 2a - a^2 n \end{aligned} \tag{II.35}$$

Las demostraciones se encuentran en [3].

Si se considera un punto en el modelo conforme $F(x) = 2x + x^2 n - \bar{n}$, para un $x \in \mathbb{R}^3$, se tiene lo siguiente:

$$\begin{aligned} T_a F(x) \tilde{T_a} &= T_a(2x)\tilde{T_a} + T_a(x^2 n)\tilde{T_a} - T_a(\bar{n})\tilde{T_a} \\ &= 2(x + n(a \cdot x)) + x^2 n - \bar{n} + 2a + a^2 n \\ &= 2x + 2n(a \cdot x) + x^2 n - \bar{n} + 2a + a^2 n \\ T_a F(x) \tilde{T_a} &= 2(x + a) + (x + a)^2 n - \bar{n} \end{aligned} \tag{II.36}$$

Lo que claramente es una traslación de x en a unidades.

2.2 Rotación

Las rotaciones en el modelo conforme siguen los mismos lineamientos que se vieron en el capítulo 1, es decir, sea $F(x) = 2x + x^2 n - \bar{n}$, la rotación de $F(x)$ se expresa como:

$$X' = RF(x)\tilde{R} \tag{II.37}$$

Donde R:

$$R = cos(\theta/2) - \hat{B}sin(\theta/2) \tag{II.38}$$

Siendo $\hat{B}$ un bivector unitario y θ el angulo de rotación.

Por ejemplo, si se quiere rotar $F(x)$ donde $x = a_1e_1 + a_2e_2 + a_3e_3$ alrededor del eje $Y = e_2$, se toma $\hat{B} = e_3e_1$:

$$X' = cos(\theta/2) - e_3e_1sin(\theta/2)F(x)cos(\theta/2) + e_3e_1sin(\theta/2) \tag{II.39}$$

Resolver esta ecuación da como resultado un $X' = 2x' + x'^2 n - \bar{n}$, donde:

$$\begin{aligned} x' =& (a_1\cos^2\alpha - a_1\sin^2\alpha + 2a_3\sin\alpha\cos\alpha)e_1 + \\ & a_2e_2 + \\ & (a_3\cos^2\alpha - a_3\sin^2\alpha - 2a_1\sin\alpha\cos\alpha)e_3 \end{aligned} \tag{II.40}$$

La demostración de esta expresión se encuentra en el apéndice A.

Además, con el uso de rotores se puede expresar la rotación y traslación como un solo rotor, es decir:

$$X' = RF(x)\tilde{R} \tag{II.41}$$

Donde,

$$R = T_aR\tilde{T_a} \quad y \quad \tilde{R} = \tilde{T}_{-a}\tilde{R}\tilde{T}_{-a} \tag{II.42}$$

Siendo a el vector de desplazamiento. La construcción de este rotor se puede encontrar en [3].

2.3 Dilatación (Escalamiento o Escalado)

Esta operación consiste en encoger o estirar un vector. Y al igual como la traslación y rotación empleamos un rotor para ello. La idea descrita por Doran y Lasenby consistía en multiplicar por $e^{-\alpha}$ de la siguiente manera:

Sea,

$$D_\alpha = e^{\frac{\alpha N}{2}} = cosh(\alpha/2) + sinh(\alpha/2)N \tag{II.43}$$

Donde $N = e\bar{e} = \frac{1}{2}\bar{n} \wedge n$. Por lo tanto:

$$F(e^{-\alpha}x) = e^{-\alpha} D_\alpha F(x) \tilde{D}_\alpha \tag{II.44}$$

De esta forma se escala un vector en el modelo conforme, por ejemplo, si se quiere escalar un vector por un factor de 3 unidades, se obtiene que:

$$e^\alpha = \frac{1}{3} \Rightarrow \alpha = ln(0{,}33) = -1{,}108. \tag{II.45}$$

Por otro lado se tiene que::

$$D_{-1,108} = cosh(-1{,}108/2) + sinh(-1{,}108/2)N \tag{II.46}$$

Por lo tanto solo se calcularía la siguiente expresión:

$$F(e^{1,108}x) = e^{1,108} D_{-1,108} F(x) \tilde{D}_{-1,108} \tag{II.47}$$

Ahora esta escala se realiza usando el origen como pivote. Si se desea escalar cualquier vector sin importar su ubicación se tendría que:

$$D_\alpha = e\left(\frac{\alpha}{2}\frac{A \wedge n}{A \cdot n}\right) \tag{II.48}$$

Donde $A = F(a)$, siendo a el vector de traslación. Los detalles de su construcción y las demostración pueden ser encontrados en [3] y [12].

2.4 Reflexión

En el capítulo I (I.4.1) se menciona que reflejar usando el álgebra geométrica tenía la siguiente representación:

$$a = -\hat{n}a\hat{n} \tag{II.49}$$

La geometría conforme presenta una representación similar, es decir, una reflexión en el modelo conforme tiene la siguiente representación [12]:

$$X' = \pi X \pi \tag{II.50}$$

Donde π es el plano en el cual se refleja, y X es el punto en el modelo conforme a reflejar. En el apéndice A se muestra la resolución de esta ecuación.

3 Intersecciones de primitivas

Una de las ventajas del modelo conforme, es la manera en como se manejan las intersecciones, se muestra que prácticamente con una sola fórmula podemos sacar información sobre la intersección de los diferentes elementos de la geometría. No se pretende demostrar o deducir dichas fórmulas, una explicación detallada se encuentra en las referencias [12] y [14].

El operador a utilizar para indicar intersección será $\vee$. Además se van a usar un máximo de 8 puntos en $\mathbb{R}^3$ que se definen de la siguiente manera:

$$\begin{aligned} x_1 &= a_1e_1 + a_2e_2 + a_3e_3, & x_5 &= q_1e_1 + q_2e_2 + q_3e_3 \\ x_2 &= b_1e_1 + b_2e_2 + d_3e_3, & x_6 &= w_1e_1 + w_2e_2 + w_3e_3 \\ x_3 &= c_1e_1 + c_2e_2 + c_3e_3, & x_7 &= p_1e_1 + p_2e_2 + p_3e_3 \\ x_4 &= d_1e_1 + d_2e_2 + d_3e_3, & x_8 &= z_1e_1 + z_2e_2 + z_3e_3 \end{aligned}$$

Y sus puntos en el modelo conforme están representados como:

$$\begin{aligned} P_1 &= 2x_1 + x_1^2 n - \bar{n}, & P_5 &= 2x_5 + x_5^2 n - \bar{n} \\ P_2 &= 2x_2 + x_2^2 n - \bar{n}, & P_6 &= 2x_6 + x_6^2 n - \bar{n} \\ P_3 &= 2x_3 + x_3^2 n - \bar{n}, & P_7 &= 2x_7 + x_7^2 n - \bar{n} \\ P_4 &= 2x_4 + x_4^2 n - \bar{n}, & P_8 &= 2x_8 + x_8^2 n - \bar{n} \end{aligned}$$

Por último se presenta una definición que se va a emplear en las intersecciones.

Definición 11 *Sea A, algún elemento del álgebra definimos el dual de A, como* A^*,

$$A^* = IA \tag{II.51}$$

3.1 Intersección Recta-Recta

Sean dos rectas, $L_1 = P_1 \wedge P_2 \wedge n$ y $L_2 = P_3 \wedge P_4 \wedge n$ la intersección viene dada por la fórmula,

$$B = (L_1 \vee L_2)^* = (IL_1) \cdot L_2 = (L_1) \cdot (IL_2) \tag{II.52}$$

Como se demuestra en el Apéndice A (A.2.1), B es igual a,

$$\begin{aligned} B =&(\alpha_2\beta_5 - \alpha_3\beta_4 + \alpha_4\beta_3 - \alpha_5\beta_2)e_1e_2e + (\alpha_3\beta_6 - \alpha_1\beta_5 + \alpha_5\beta_1 - \alpha_6\beta_3)e_2e_3e \\ &+ (\alpha_1\beta_4 - \alpha_2\beta_6 + \alpha_6\beta_2 - \alpha_4\beta_1)e_3e_1e + (\alpha_2\beta_5 - \alpha_3\beta_4 + \alpha_4\beta_3 - \alpha_5\beta_2)e_1e_2\bar{e} \\ &+ (\alpha_3\beta_6 - \alpha_1\beta_5 + \alpha_5\beta_1 - \alpha_6\beta_3)e_2e_3\bar{e} + (\alpha_1\beta_4 - \alpha_2\beta_6 + \alpha_6\beta_2 - \alpha_4\beta_1)e_3e_1\bar{e} \\ &+ (\alpha_6\beta_5 - \alpha_5\beta_6)e_1e\bar{e} + (\alpha_4\beta_6 - \alpha_6\beta_4)e_2e\bar{e} + (\alpha_5\beta_4 - \alpha_4\beta_5)e_3e\bar{e} \\ &+ (\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1)e \\ &+ (\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1)\bar{e} \end{aligned} \tag{II.53}$$

Donde α y β toman los valores,

$$\begin{aligned} \alpha_1 &= (a_1b_2 - a_2b_1), & \beta_1 &= (c_1d_2 - c_2d_1) \\ \alpha_2 &= (a_2b_3 - a_3b_2), & \beta_2 &= (c_2d_3 - c_3d_2) \\ \alpha_3 &= (a_3b_1 - a_1b_3), & \beta_3 &= (c_3d_1 - c_1d_3) \\ \alpha_4 &= (a_1 - b_1), & \beta_4 &= (c_1 - d_1) \\ \alpha_5 &= (a_2 - b_2), & \beta_5 &= (c_2 - d_2) \\ \alpha_6 &= (a_3 - b_3), & \beta_6 &= (c_3 - d_3) \end{aligned}$$

De la expresión II.53, el valor que determina la existencia de una intersección es,

$$(\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1) \tag{II.54}$$

Si este valor es igual a 0, implica que las dos rectas se intersectan.

3.2 Intersección Recta-Plano

Sea la recta $L = P_1 \wedge P_2 \wedge n$ y el plano $\Pi = P_3 \wedge P_4 \wedge P_5 \wedge n$ la intersección viene determinada por,

$$B = (\Pi \vee L) = (I\Pi) \cdot L \tag{II.55}$$

Como se demuestra en el apéndice A (A.2.2), B es la siguiente expresión,

$$\begin{aligned} B =&(\omega_2\beta_3 + \omega_1\beta_4 - \omega_4\beta_1)e_1e + (\omega_2\beta_3 + \omega_1\beta_4 - \omega_4\beta_1)e_1\bar{e} \\ &(\omega_3\beta_1 - \omega_2\beta_2 + \omega_1\beta_5)e_2e + (\omega_3\beta_1 - \omega_2\beta_2 - \omega_1\beta_5)e_2\bar{e} \\ &(-\omega_3\beta_3 + \omega_4\beta_2 + \omega_1\beta_6)e_3e + (-\omega_3\beta_3 + \omega_4\beta_2 + \omega_1\beta_6)e_3\bar{e} \\ &(-\omega_3\beta_4 - \omega_4\beta_5 - \omega_2\beta_6)e\bar{e} \end{aligned} \tag{II.56}$$

Donde β y ω son iguales a,

$$\begin{aligned}
\beta_1 &= (a_1 b_2 - a_2 b_1) \\
\beta_2 &= (a_2 b_3 - a_3 b_2) \\
\beta_3 &= (a_3 b_1 - a_1 b_3) \\
\beta_4 &= (a_1 - b_1) \\
\beta_5 &= (a_2 - b_2) \\
\beta_6 &= (a_3 - b_3)
\end{aligned} \tag{II.57}$$

$$\begin{aligned}
\omega_1 &= (c_1 d_2 q_3 + c_2 d_3 q_1 + c_3 d_1 q_2 - c_2 d_1 q_3 - c_3 d_2 q_1 - c_1 d_3 q_2) \\
\omega_2 &= (c_1 d_2 - c_2 d_1 - c_1 q_2 + c_2 q_1 + d_1 q_2 - d_2 q_1) \\
\omega_3 &= (c_2 d_3 - c_3 d_2 - c_2 q_3 + c_3 q_2 + d_2 q_3 - d_3 q_2) \\
\omega_4 &= (c_3 d_1 - c_1 d_3 - c_3 q_1 + c_1 q_3 + d_3 q_1 - d_1 q_3)
\end{aligned} \tag{II.58}$$

Calculando B^2 se obtiene información sobre la intersección,

$$B^2 = (\omega_3 \beta_4 + \omega_4 \beta_5 + \omega_2 \beta_6)^2 \tag{II.59}$$

Esta última expresión tiene dos posibles casos:

1. Si $B = 0$, implica que la recta es paralela al plano, pero hay dos opciones, que la recta este contenida en el plano o sea paralela al mismo, para determinar esto último, se toma la expresión $(-\omega_3 \beta_3 + \omega_4 \beta_2 + \omega_1 \beta_6)$ si esta expresión es igual a 0, entonces la recta vive en el plano, de lo contrario la recta no reside en el plano.

2. Si $B > 0$, entonces la recta intersecta al plano en un punto.

3.3 Intersección Recta-Esfera

Sea la recta $L = P_1 \wedge P_2 \wedge n$, y la esfera $E = P_3 \wedge P_4 \wedge P_5 \wedge P_6$ la intersección se define por,

$$B = (E \vee L) = (IE) \cdot L \tag{II.60}$$

Como se demuestra en el Apéndice A (A.2.3), B viene dado por,

$$\begin{aligned}
B =& (\mu_3 \beta_3 + \mu_1 \beta_4 - \mu_5 \beta_1) e_1 e + (\mu_2 \beta_4 + \mu_3 \beta_3 - \mu_5 \beta_1) e_1 \bar{e} \\
&+ (\mu_4 \beta_1 + \mu_1 \beta_5 - \mu_3 \beta_2) e_2 e + (\mu_4 \beta_1 + \mu_2 \beta_5 - \mu_3 \beta_2) e_2 \bar{e} \\
&+ (\mu_5 \beta_2 + \mu_1 \beta_6 - \mu_4 \beta_3) e_3 e + (\mu_5 \beta_2 + \mu_2 \beta_6 - \mu_4 \beta_3) e_3 \bar{e} \\
&+ (\mu_2 \beta_1 - \mu_1 \beta_1) e_1 e_2 + (\mu_2 \beta_2 - \mu_1 \beta_2) e_2 e_3 + (\mu_2 \beta_3 - \mu_1 \beta_3) e_3 e_1 \\
&- (\mu_4 \beta_4 + \mu_5 \beta_5 + \mu_3 \beta_6) e \bar{e}
\end{aligned} \tag{II.61}$$

Donde β es igual a las expresiones (II.57) y μ es igual a,

$$\begin{aligned}
\mu_1 =&\alpha_{124}(1-x_5^2)+\alpha_{234}(1-x_3^2)+\alpha_{314}(1-x_4^2)+\alpha_{123}(x_6^2-1)\\
\mu_2 =&\alpha_{123}(1+x_6^2)-\alpha_{124}(1+x_5^2)-\alpha_{234}(1+x_3^2)-\alpha_{314}(1+x_4^2)\\
\mu_3 =&ad_{12}(x_5^2-x_4^2)+bd_{12}(x_3^2-x_5^2)+cd_{12}(x_4^2-x_3^2)+ab_{12}(x_6^2-x_5^2)\\
&+bc_{12}(x_6^2-x_3^2)+ca_{12}(x_6^2-x_4^2)\\
\mu_4 =&ad_{23}(x_5^2-x_4^2)+bd_{23}(x_3^2-x_5^2)+cd_{23}(x_4^2-x_3^2)+ab_{23}(x_6^2-x_5^2)\\
&+bc_{23}(x_6^2-x_3^2)+ca_{23}(x_6^2-x_4^2)\\
\mu_5 =&ad_{31}(x_5^2-x_4^2)+bd_{31}(x_3^2-x_5^2)+cd_{31}(x_4^2-x_3^2)+ab_{31}(x_6^2-x_5^2)\\
&+bc_{31}(x_6^2-x_3^2)+ca_{31}(x_6^2-x_4^2)
\end{aligned} \tag{II.62}$$

Donde,

$$\begin{aligned}
\alpha_{124} &= (c_1d_2w_3+c_2d_3w_1+c_3d_1w_2-c_2d_1w_3-c_3d_2w_1-c_1d_3w_2)\\
\alpha_{234} &= (d_1w_2w_3+d_2q_3w_1+d_3q_1w_2-d_2q_1w_3-d_3q_2w_1-d_1q_3w_2)\\
\alpha_{314} &= (w_1c_2w_3+q_2c_3w_1+q_3c_1w_2-q_2c_1w_3-q_3c_2w_1-q_1c_3w_2)\\
\alpha_{123} &= (c_1d_2q_3+c_2d_3q_1+c_3d_1q_2-c_2d_1q_3-c_3d_2q_1-c_1d_3q_2)
\end{aligned} \tag{II.63}$$

Y las variables del tipo ad_{12}, bc_{23}, etc, se expresan con la siguiente notación,

$$xy_{ij} = x_{i+2}y_{j+2} - x_{j+2}y_{i+2}$$

Calculando B^2 se obtiene,

$$\begin{aligned}
B^2 =& -(\mu_3\beta_3+\mu_1\beta_4-\mu_5\beta_1)^2+(\mu_2\beta_4+\mu_3\beta_3-\mu_5\beta_1)^2\\
&-(\mu_4\beta_1+\mu_1\beta_5-\mu_3\beta_2)^2+(\mu_4\beta_1+\mu_2\beta_5-\mu_3\beta_2)^2\\
&+(\mu_5\beta_2+\mu_1\beta_6-\mu_4\beta_3)^2+(\mu_5\beta_2+\mu_2\beta_6-\mu_4\beta_3)^2\\
&-(\mu_2\beta_1-\mu_1\beta_1)^2-(\mu_2\beta_2-\mu_1\beta_2)^2-(\mu_2\beta_3-\mu_1\beta_3)^2\\
&+(\mu_4\beta_4+\mu_5\beta_5+\mu_3\beta_6)^2
\end{aligned} \tag{II.64}$$

Esta última expresión tiene tres posibles casos:

1. Si $B^2 = 0$, la recta es tangente a la esfera.
2. Si $B^2 > 0$, la recta intersecta a la esfera en dos puntos.
3. Si $B^2 < 0$, la recta no intersecta a la esfera.

3.4 Intersección Plano-Plano

Sean dos planos $\Pi_1 = P_1 \wedge P_2 \wedge P_3 \wedge n$ y $\Pi_2 = P_4 \wedge P_5 \wedge P_6 \wedge n$, la intersección se define por,

$$B = (\Pi_1 \vee \Pi_2) = (I\Pi_1) \cdot \Pi_2 \tag{II.65}$$

Como se demuestra en el Apéndice A (A.2.4), B es la siguiente expresión,

$$\begin{aligned} B =&(\omega_4\lambda_2 - \omega_2\lambda_4)e_1e\bar{e} + (\omega_2\lambda_3 - \omega_3\lambda_2)e_2e\bar{e} + (\omega_3\lambda_4 - \omega_4\lambda_3)e_3e\bar{e} \\ &(\omega_2\lambda_1 - \omega_1\lambda_2)e_1e_2e + (\omega_3\lambda_1 - \omega_1\lambda_3)e_2e_3e + (\omega_4\lambda_1 - \omega_1\lambda_4)e_3e_1e \\ &(\omega_2\lambda_1 - \omega_1\lambda_2)e_1e_2\bar{e} + (\omega_3\lambda_1 - \omega_1\lambda_3)e_2e_3\bar{e} + (\omega_4\lambda_1 - \omega_1\lambda_4)e_3e_1\bar{e} \end{aligned} \tag{II.66}$$

Donde ω y λ son iguales a,

$$\begin{aligned} \omega_1 &= (a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_2b_1c_3 - a_3b_2c_1 - a_1b_3c_2) \\ \omega_2 &= (a_1b_2 - a_2b_1 - a_1c_2 + a_2c_1 + b_1c_2 - b_2c_1) \\ \omega_3 &= (a_2b_3 - a_3b_2 - a_2c_3 + a_3c_2 + b_2c_3 - b_3c_2) \\ \omega_4 &= (a_3b_1 - a_1b_3 - a_3c_1 + a_1c_3 + b_3c_1 - b_1c_3) \end{aligned} \tag{II.67}$$

$$\begin{aligned} \lambda_1 &= (d_1q_2w_3 + d_2q_3w_1 + d_3q_1w_2 - d_2q_1w_3 - d_3q_2w_1 - d_1q_3w_2) \\ \lambda_2 &= (d_1q_2 - d_2q_1 - d_1w_2 + d_2w_1 + q_1w_2 - q_2w_1) \\ \lambda_3 &= (d_2q_3 - d_3q_2 - d_2w_3 + d_3w_2 + q_2w_3 - q_3w_2) \\ \lambda_4 &= (d_3q_1 - d_1q_3 - d_3w_1 + d_1w_3 + q_3w_1 - q_1w_3) \end{aligned} \tag{II.68}$$

Calculando B^2,

$$B^2 = (\omega_4\lambda_2 - \omega_2\lambda_4)^2 + (\omega_2\lambda_3 - \omega_3\lambda_2)^2 + (\omega_3\lambda_4 - \omega_4\lambda_3)^2 \tag{II.69}$$

Se obtienen dos posibles casos,

1. Si $B = 0$, los planos son paralelos.
2. Si $B > 0$, los planos se intersectan

3.5 Intersección Esfera-Plano

Sea una esfera $E = P_1 \wedge P_2 \wedge P_3 \wedge P_4$ y un plano $\Pi = P_5 \wedge P_6 \wedge P_7 \wedge n$, la intersección se define por,

$$B = (E \vee \Pi) = (IE) \cdot \Pi \tag{II.70}$$

Como se demuestra en el Apéndice A (A.2.5), B se expresa como,

$$\begin{aligned} B =&(\mu_5\omega_2 - \mu_3\omega_4)e_1e\bar{e} + (\mu_3\omega_3 - \mu_4\omega_2)e_2e\bar{e} + (\mu_4\omega_4 - \mu_5\omega_3)e_3e\bar{e} \\ &+ (\mu_3\omega_1 - \mu_1\omega_2)e_1e_2e + (\mu_4\omega_1 - \mu_1\omega_3)e_2e_3e + (\mu_5\omega_1 - \mu_1\omega_4)e_3e_1e \\ &+ (\mu_3\omega_1 - \mu_2\omega_2)e_1e_2\bar{e} + (\mu_4\omega_1 - \mu_2\omega_3)e_2e_3\bar{e} + (\mu_5\omega_1 - \mu_2\omega_4)e_3e_1\bar{e} \\ &+ (\mu_1\omega_1 - \mu_2\omega_1)e_1e_2e_3 \end{aligned} \tag{II.71}$$

Donde μ y ω son iguales a,

$$\begin{aligned} \mu_1 =&\alpha_{124}(1 - x_3^2) + \alpha_{234}(1 - x_1^2) + \alpha_{314}(1 - x_2^2) + \alpha_{123}(x_4^2 - 1) \\ \mu_2 =&\alpha_{123}(1 + x_4^2) - \alpha_{124}(1 + x_3^2) - \alpha_{234}(1 + x_1^2) - \alpha_{314}(1 + x_2^2) \\ \mu_3 =&ad_{12}(x_3^2 - x_2^2) + bd_{12}(x_1^2 - x_3^2) + cd_{12}(x_2^2 - x_1^2) + ab_{12}(x_4^2 - x_3^2) \\ &+ bc_{12}(x_4^2 - x_1^2) + ca_{12}(x_4^2 - x_2^2) \\ \mu_4 =&ad_{23}(x_3^2 - x_2^2) + bd_{23}(x_1^2 - x_3^2) + cd_{23}(x_2^2 - x_1^2) + ab_{23}(x_4^2 - x_3^2) \\ &+ bc_{23}(x_4^2 - x_1^2) + ca_{23}(x_4^2 - x_2^2) \\ \mu_5 =&ad_{31}(x_3^2 - x_2^2) + bd_{31}(x_1^2 - x_3^2) + cd_{31}(x_2^2 - x_1^2) + ab_{31}(x_4^2 - x_3^2) \\ &+ bc_{31}(x_4^2 - x_1^2) + ca_{31}(x_4^2 - x_2^2) \end{aligned} \tag{II.72}$$

Donde,

$$\begin{aligned} \alpha_{124} &= (a_1b_2d_3 + a_2b_3d_1 + a_3b_1d_2 - a_2b_1d_3 - a_3b_2d_1 - a_1b_3d_2) \\ \alpha_{234} &= (b_1c_2d_3 + b_2c_3d_1 + b_3c_1d_2 - b_2c_1d_3 - b_3c_2d_1 - b_1c_3d_2) \\ \alpha_{314} &= (c_1a_2d_3 + c_2a_3d_1 + c_3a_1d_2 - c_2a_1d_3 - c_3a_2d_1 - c_1a_3d_2) \\ \alpha_{123} &= (a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_2b_1c_3 - a_3b_2c_1 - a_1b_3c_2) \end{aligned} \tag{II.73}$$

Y las variables del tipo ad_{12}, bc_{23}, etc, se expresan con la siguiente notación,

$$xy_{ij} = x_iy_j - x_jy_i \tag{II.74}$$

Por último,

$$\begin{aligned} \omega_1 &= (q_1w_2p_3 + q_2w_3p_1 + q_3w_1p_2 - q_2w_1p_3 - q_3w_2p_1 - q_1w_3p_2) \\ \omega_2 &= (q_1w_2 - q_2w_1 - q_1p_2 + q_2p_1 + w_1p_2 - w_2p_1) \\ \omega_3 &= (q_2w_3 - q_3w_2 - q_2p_3 + q_3p_2 + w_2p_3 - w_3p_2) \\ \omega_4 &= (q_3w_1 - q_1w_3 - q_3p_1 + q_1p_3 + w_3p_1 - w_1p_3) \end{aligned} \tag{II.75}$$

Calculando B^2 se obtiene,

$$\begin{aligned} B^2 =&(\mu_5\omega_2 - \mu_3\omega_4)^2 + (\mu_3\omega_3 - \mu_4\omega_2)^2 + (\mu_4\omega_4 - \mu_5\omega_3)^2 \\ &- (\mu_3\omega_1 - \mu_1\omega_2)^2 - (\mu_4\omega_1 - \mu_1\omega_3)^2 - (\mu_5\omega_1 - \mu_1\omega_4)^2 \\ &+ (\mu_3\omega_1 - \mu_2\omega_2)^2 + (\mu_4\omega_1 - \mu_2\omega_3)^2 + (\mu_5\omega_1 - \mu_2\omega_4)^2 - (\mu_1\omega_1 - \mu_2\omega_1)^2 \end{aligned} \tag{II.76}$$

De esta última expresión se tiene tres posibles casos:

1. Si $B^2 = 0$, el plano es tangente a la esfera.
2. Si $B^2 > 0$, el plano intersecta a la esfera.
3. Si $B^2 < 0$, el plano no intersecta a la esfera.

3.6 Intersección Esfera-Esfera

Sean dos esferas $E_1 = P_1 \wedge P_2 \wedge P_3 \wedge P_4$ y $E_2 = P_5 \wedge P_6 \wedge P_7 \wedge P_8$, la intersección se define,

$$B = (E_1 \vee E_2) = (IE_1) \cdot E_2 \tag{II.77}$$

Como se demuestra en el Apéndice A (A.2.5), B se expresa como,

$$\begin{aligned} B =&(\mu_3\lambda_1 - \mu_1\lambda_3)e_1e_2e + (\mu_4\lambda_1 - \mu_1\lambda_4)e_2e_3e + (\mu_5\lambda_1 - \mu_1\lambda_5)e_3e_1e \\ &+ (\mu_3\lambda_2 - \mu_2\lambda_3)e_1e_2\bar{e} + (\mu_4\lambda_2 - \mu_2\lambda_4)e_2e_3\bar{e} + (\mu_5\lambda_2 - \mu_2\lambda_5)e_3e_1\bar{e} \\ &+ (\mu_5\lambda_3 - \mu_3\lambda_5)e_1e\bar{e} + (\mu_3\lambda_4 - \mu_4\lambda_3)e_2e\bar{e} + (\mu_4\lambda_5 - \mu_5\lambda_4)e_3e\bar{e} \\ &+ (\mu_1\lambda_2 - \mu_2\lambda_1)e_1e_2e_3 \end{aligned} \tag{II.78}$$

Donde μ toma los mismos valores que las expresiones (II.62), y λ es igual a,

$$\begin{aligned} \lambda_1 =&\alpha_{124}(1 - x_7^2) + \alpha_{234}(1 - x_5^2) + \alpha_{314}(1 - x_6^2) + \alpha_{123}(x_7^2 - 1) \\ \lambda_2 =&\alpha_{123}(1 + x_8^2) - \alpha_{124}(1 + x_7^2) - \alpha_{234}(1 + x_5^2) - \alpha_{314}(1 + x_6^2) \\ \lambda_3 =&ad_{12}(x_7^2 - x_6^2) + bd_{12}(x_5^2 - x_7^2) + cd_{12}(x_6^2 - x_5^2) + ab_{12}(x_8^2 - x_7^2) \\ &+ bc_{12}(x_8^2 - x_5^2) + ca_{12}(x_8^2 - x_6^2) \\ \lambda_4 =&ad_{23}(x_7^2 - x_6^2) + bd_{23}(x_5^2 - x_7^2) + cd_{23}(x_6^2 - x_5^2) + ab_{23}(x_8^2 - x_7^2) \\ &+ bc_{23}(x_8^2 - x_5^2) + ca_{23}(x_8^2 - x_6^2) \\ \lambda_5 =&ad_{31}(x_7^2 - x_6^2) + bd_{31}(x_5^2 - x_7^2) + cd_{31}(x_6^2 - x_5^2) + ab_{31}(x_8^2 - x_7^2) \\ &+ bc_{31}(x_8^2 - x_5^2) + ca_{31}(x_8^2 - x_6^2) \end{aligned} \tag{II.79}$$

Donde,

$$\begin{aligned} \alpha_{124} &= (q_1w_2z_3 + q_2w_3z_1 + q_3w_1z_2 - q_2w_1z_3 - q_3w_2z_1 - q_1w_3z_2) \\ \alpha_{234} &= (w_1p_2z_3 + w_2p_3z_1 + w_3p_1z_2 - w_2p_1z_3 - w_3p_2z_1 - w_1p_3z_2) \\ \alpha_{314} &= (p_1q_2z_3 + p_2q_3z_1 + p_3q_1z_2 - p_2q_1z_3 - p_3q_2z_1 - p_1q_3z_2) \\ \alpha_{123} &= (q_1w_2p_3 + q_2w_3p_1 + q_3w_1p_2 - q_2w_1p_3 - q_3w_2p_1 - q_1w_3p_2) \end{aligned} \tag{II.80}$$

Y las variables del tipo ad_{12}, bc_{23}, etc, se expresan con la siguiente notación,

$$xy_{ij} = x_{i+4}y_{j+4} - x_{j+4}y_{i+4} \tag{II.81}$$

Calculando B^2 se obtiene,

$$\begin{aligned} B^2 = &- (\mu_3\lambda_1 - \mu_1\lambda_3)^2 - (\mu_4\lambda_1 - \mu_1\lambda_4)^2 - (\mu_5\lambda_1 - \mu_1\lambda_5)^2 \\ &+ (\mu_3\lambda_2 - \mu_2\lambda_3)^2 + (\mu_4\lambda_2 - \mu_2\lambda_4)^2 + (\mu_5\lambda_2 - \mu_2\lambda_5)^2 \\ &+ (\mu_5\lambda_3 - \mu_3\lambda_5)^2 + (\mu_3\lambda_4 - \mu_4\lambda_3)^2 + (\mu_4\lambda_5 - \mu_5\lambda_4)^2 \\ &- (\mu_1\lambda_2 - \mu_2\lambda_1)^2 \end{aligned} \tag{II.82}$$

Esta última expresión tiene tres posibles casos:

1. Si $B = 0$, las esferas son tangentes.
2. Si $B > 0$, las esferas se intersectan.
3. Si $B < 0$, no hay intersección en las esferas.

Como nota final, cabe mencionar que en este trabajo no se realizan el cálculo de los puntos, rectas o círculos de intersección, ya que no son de interés para resolver los problemas que se van a tratar. No obstante, una solución para encontrar las primitivas producto de la intersección pueden ser encontrada en la referencia [12].

CAPÍTULO III

COMPUTACIÓN PARALELA Y COLISIONES

1 Computación Paralela

Algunos problemas en la computación requieren de mucho cálculo, un solo procesador tomando en cuenta el más rápido en la actualidad, podría tardar años en realizar algunas operaciones. Es aquí, donde la computación paralela toma interés, con esta técnica un problema puede ser resuelto por más de un CPU (Central Processor Unit), optimizando así el tiempo de respuesta.

1.1 Historia

La computación paralela ha tenido interés desde los años 50's hasta el dia de hoy, en la actualidad existen computadores que tienen mas de un CPU que comparten una memoria (*shared memory*), es decir, todos los procesadores tienen acceso a la misma información. No obstante, son máquinas relativamente grandes y costosas que solamente grandes compañías, gobierno, o instituciones científicas pueden obtener.

Pero recientemente ha surgido un nuevo tipo de parelelismo que es accesible a cualquier persona que posea algún computador personal. Las empresas fabricantes de procesadores se dieron cuenta que la solución para realizar cálculos más rápido no era incrementar la velocidad de los proceadores, ya que este incremento tiene la desventaja de producir demasiado calor.

Observando esto, se dieron cuenta que tenían que optimizar los procesadores de otra manera. Es en el 2002 cuando INTEL presenta una familia de procesadores con la tecnología de HyperThreading [15] (figura III.1) el más conocido de ellos es el Pentium 4. Estos procesadores podían ejecutar 2 hilos (threads) o procesos al mismo tiempo, dando una idea de paralelismo en un solo procesador.

Este paralelismo era relativo, por que aunque un procesador podía ejecutar 2 hilos al mismo tiempo, el CPU seguía ejecutando una instrucción a la vez. Solo que la diferencia del procesador sin la tecnología hyperthread, es que este ejecutaba un thread primero y el otro después. Esto brindo una eficiencia en términos de cómputo sin aumentar la velocidad del procesador, y además estos procesadores empiezan a ser accesibles en los computadores personales.

En el año 2001 IBM se encontraba trabajando con una tecnología que se le llama multi-core (multi-núcleo) creando el chip POWER4, esta tecnología consistía en un solo procesador pero que en su interior tenía 2 núcleos, a diferencia del Hyperthreading, con

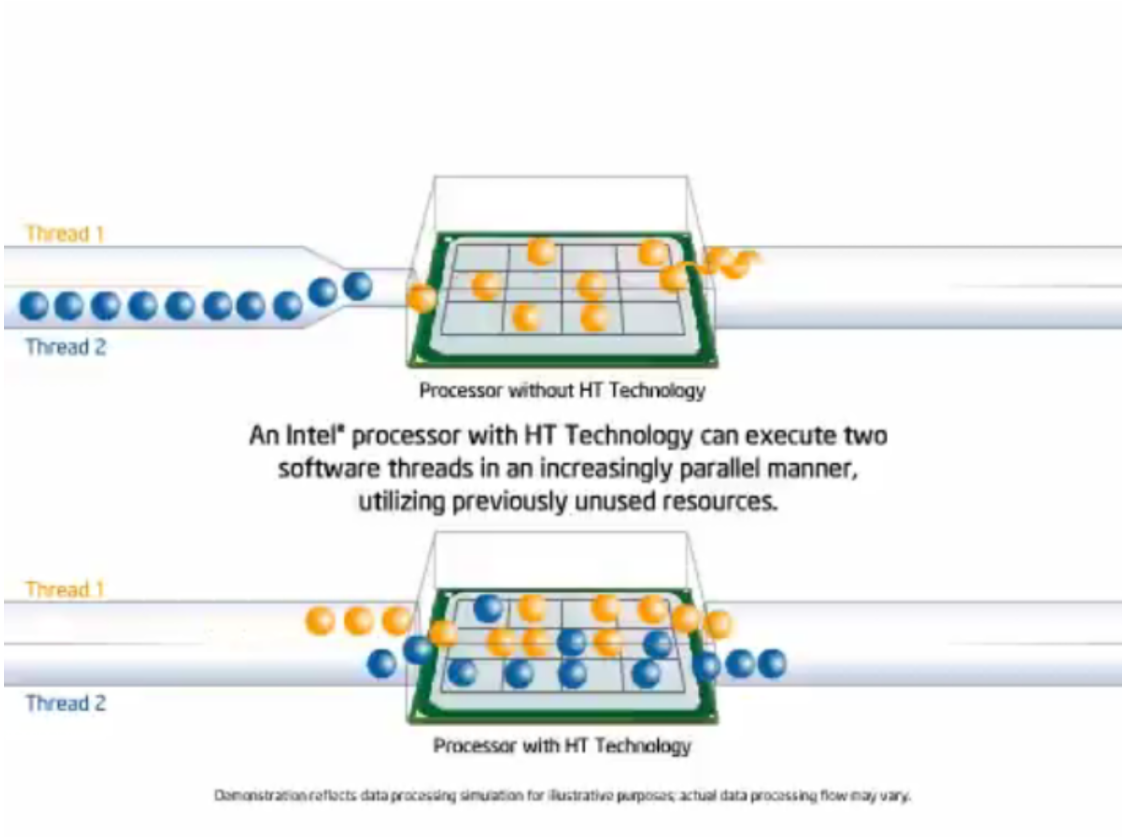


Figura III.1: *Tecnología Hyperthread [16]*

esta tecnología 2 hilos podían ejecutarse simultáneamente en forma paralela, ya que cada hilo se ejecutaba en un núcleo (core).

Esto dió inicio a una competencia fuerte sobre el mercado de los procesadores duales, mas aún en la actualidad no solamente un procesador posee dos núcleos, existen *quad-cores* (cuatro núcleos), *eight-cores* (ocho núcleos), etc, con esto se incrementaba el poder de cómputo y la rapidez sin tener que incrementar la velocidad del CPU.

Como se muestra en la figura III.2, se puede ver que ahora 2 hilos podían entrar simultáneamente al procesador, ya que cada hilo se ejecutaba en un núcleo independiente. Es aquí cuando se empieza a tener lo que se llamo un verdadero paralelismo.

Esto coloca el paralelismo en las máquinas de escritorio, y a su vez las empresas diseñadores de software empiezan a optimizar sus programas para esta nueva familia de procesadores, no obstante, este paralelismo en las computadores personales se usa mas que todo para correr programas en forma simultánea, es decir, se utiliza mayormente para procesos independientes. Pero antes de que las empresas empezaran a usar esta tecnología en sus aplicaciones, surge de manera casi inmediata una nueva visión de paralelismo, que es usando el GPU (Graphics Processor Unit), con el GPU las empresas y usuarios se dan cuenta que los cálculos son muchos más rápidos, por lo tanto, empiezan a optimizar los

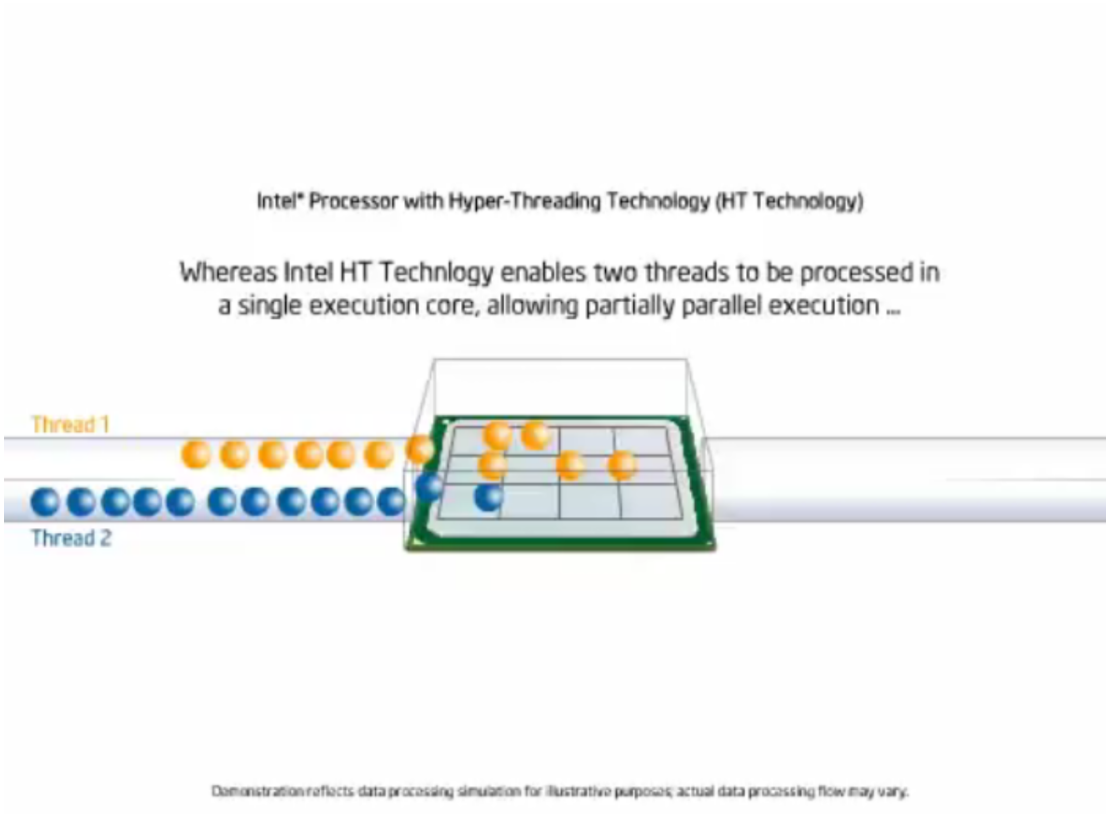


Figura III.2: *Tecnología DualCore [17]*

cálculos de los software usando el GPU, y dejan el paralelismo en el CPU para procesos independientes.

2 GPU (Graphics Processor Unit)

Desde que se empieza a utilizar la computadora con fines gráficos, las empresas creadoras del hardware se dan cuenta que el CPU no era suficiente para suplir la demanda gráfica que se requería. Un usuario, necesitaba una respuesta casi inmediata si este solicitaba algo tan simple como trasladar una geometría en 3D. Cabe mencionar, que el CPU es el cerebro de la computadora, este no solo se encarga de los cálculos, sino que además se encarga de sincronizar todo el hardware de la computadora, algo que se hace a través del sistema operativo (OS).

Silicon Graphics fue una de las empresas que se encargo de crear computadoras optimizadas para el procesamiento de gráficos 2D/3D (figura III.3), mas aún, en el año 1992 crean una librería OpenGL [18], que se convertiría en un estándar para la programación de aplicaciones 2D/3D. No obstante en aquel entonces, estas computadoras eran extremadamente costosas, al punto que solamente grandes compañías podían comprar dicho computadores. La computación gráfica de ese entonces era enfocada mayormente al mundo científico.

Figura III.3: *Sillicon Graphics - Indy [19]*

Pero en los 90's, las computadoras personales empiezan a expandirse en cada hogar del mundo, además empieza una industria que en la actualidad es muy lucrativa, que es la industria de los vídeo juegos. En los mismos 90's nacen empresas como S3 [20], 3Dfx [21], NVIDIA [22], entre otras, y se empiezan a crear tarjetas aceleradoras gráficas para los computadores personales y consolas de vídeo juegos. Haciendo que la tecnología 2D/3D llegara a cualquier computador.

Pero es en 1995 cuando Microsoft lanza el sistema operativo Windows 95, este sistema operativo (cuya interfaz era gráfica) invade de manera rápida las computadoras de escritorio, aumentado así la necesidad de aplicaciones gráficas. Microsoft consciente de esto, lanza DirectX 1.0 [23] que al igual que OpenGL era una librería que podían usar los diseñadores de software para crear aplicaciones 2D/3D. Estas librerías (OpenGL/DirectX) usaban el potencial de las tarjetas gráficas para correr aplicaciones en tiempo real, como por ejemplo, los vídeos juegos.

Estas librerías presentaban un problema, los programadores estaban restringidos a lo que las librerías gráficas proporcionaban, si un programador de software 2D/3D quería implementar algún algoritmo gráfico usando el GPU, tan sencillamente no podía o tenían que ser un experto usando lenguajes como *assembly*, los programadores entonces hacían los cálculos en el CPU y luego pasaban la información obtenida al GPU usando las funciones de OpenGL o DirectX.

En los mismos 90's, las empresas fabricantes de procesadores, viendo el auge y la necesidad de optimizar los cálculos en 3D, empiezan a optimizar los CPU. Se implementa en los CPU una manera de realizar cálculos en paralelos, usando SIMD (*Single instruction,*

multiple data / simple instrucción, multiple información) [24], es decir, esta técnica consiste en que se toma un grupo de información de la memoria y se le aplica un cálculo simple. Esto era algo muy útil en la computación gráfica y aplicaciones multimedias, ya que la mayoría de los cálculos en estas áreas, implican suma de vectores, multiplicación por escalar, entre otras operaciones. A continuación se muestra un ejemplo:

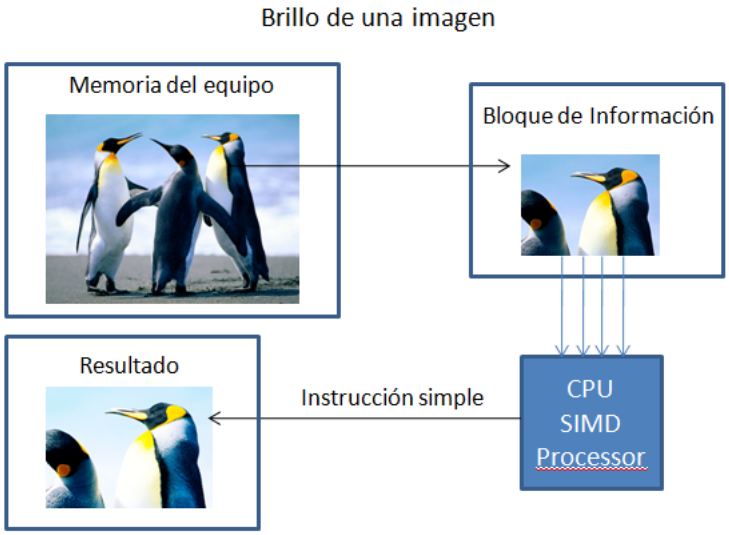


Figura III.4: *SIMD (Single instruction, multiple data)*

Como se puede ver en la figura III.4, el SIMD, funciona de la siguiente manera, en primera instancia se toma un bloque de información de la memoria, y a este bloque de información se le suma un valor para aumentar el brillo de la imagen, el CPU a través del procesador SIMD, aplica este cálculo de manera simultánea a todo el bloque de información, y regresaba el resultado a la memoria. Esta era una forma de paralelizar cálculos. El Pentium MMX fue uno de los primeros procesadores Intel en implementar esta tecnología.

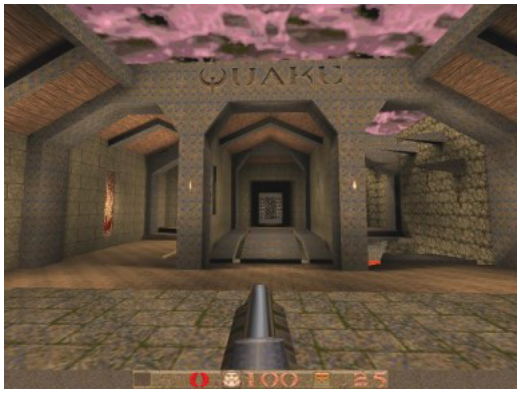


Figura III.5: *Id-Software - Quake [25]*

Para aquel entonces, el mundo de los vídeos juegos estaba surgiendo, juegos como QUAKE (figura III.5) lanzado el 22 de Junio de 1996, fue uno de los primeros juegos

en primera persona en usar geometría 3D (polígonos), además, también fue uno de los primeros en empezar a utilizar los aceleradores gráficos para presentar un gran acabado en su arte. A mediados de los 90's, la industria de los vídeo juegos crecía de manera muy rápida, se necesitaban de procesadores gráficos cada vez más potente, para presentar mejor calidad gráfica en los vídeos juegos. Es entonces, cuando las empresas encargadas del hardware y del software (OpenGL y DirectX) toman medidas al respecto, y empiezan a proporcionarle al programador la capacidad de añadir sus propios códigos usando el GPU del equipo.

2.1 Vertex Shader y Fragment/Pixel Shader

En líneas generales OpenGL y DirectX siguen él siguiente esquema de trabajo (figura III.6) para presentar una geometría 3D en la pantalla de un computador .

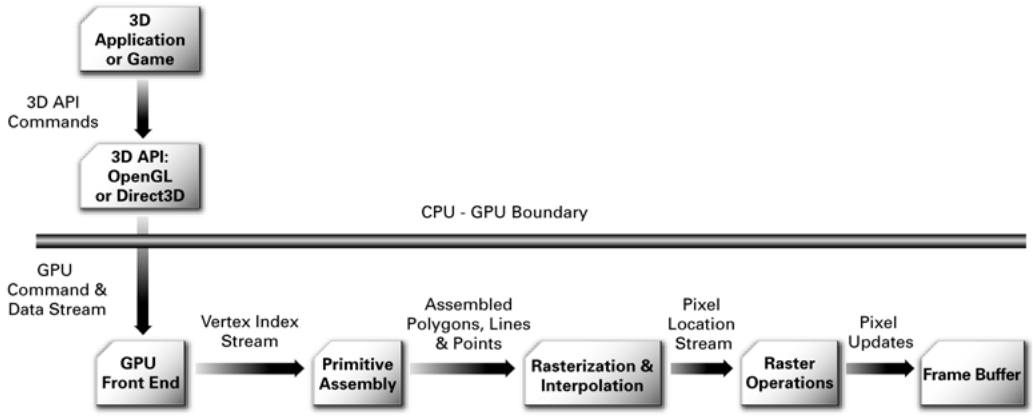


Figura III.6: *Pipeline gráfico [26]*

Las primeras dos fases consiste en la aplicación (3D Application or Game) y en las funciones gráficas (3D API), estas son las líneas de códigos que el programador implementa. Una vez ejecutado el programa, se procede a lo siguiente:

- **Procesamiento de los vértices** (GPU Front End) y (*Primitive Assembly*): en esta fase la tarjeta gráfica toma los vértices suministrado por el programador, y le aplica varias transformaciones, como por ejemplo, traslación, rotación, etc. Luego se calcula la iluminación por cada vértice usando las normales suministradas por el programador, y después se procede a proyectar en el plano de la cámara, es aquí donde se hace un proceso de clipping, el cual consiste en cortar aquellos triángulos que intersectan el campo de visión y omitir los triángulos no visibles.

- **Procesamiento de píxeles** (Rasterization and Interpolation) y (Raster Operation): en estas dos fases se procede a llevar los triángulos a píxeles, es decir, se convierte una línea, en coordenadas de píxeles para dibujarla, además se rellenan los píxeles que contiene un triángulo, y también en esta fase se aplica la textura en caso de que esta exista y se halla enlazado a la geometría.

- **FrameBuffer**: el *frame buffer* es un espacio de memoria en la tarjeta gráfica donde se almacenan los píxeles que van a ser visibles en la ventana de la aplicación.

Básicamente el proceso se divide en dos: vértices y píxeles, mayor información se puede encontrar en la referencia [27]. En los inicios de las tarjetas gráficas el programador no podía alterar, modificar o cambiar, algo en estos procesos, los programadores estaban restringidos a lo que hacia la tarjeta gráfica. Es decir, si el programador estaba interesado en aplicar algún algoritmo de iluminación, o algún efecto en los píxeles, tan sencillamente no podía, y los hacia en el CPU.

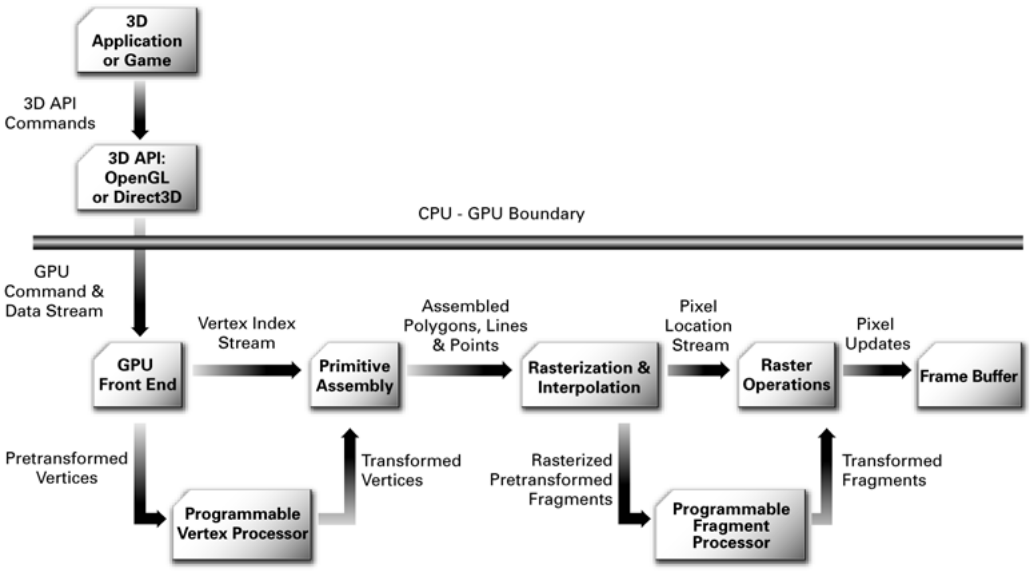


Figura III.7: *Vertex Shader y Pixel Shader Pipeline [26]*

Pero es a partir del año 2000 que empieza un cambio radical en el pipeline gráfico, las tarjetas gráficas y las librerías de OpenGL y DirectX empiezan a darle mas poder al programador para usar el GPU. NVIDIA saca al mercado la serie Geforce 3 y ATI introduce la ATI Radeon 9700, estas tarjetas eran compatible con OpenGL 1.2 y Direct3D 9.0, y aquí las librerías gráficas introducen el lenguaje de *shader* (*Shaders Language*), esto permitía al programador mejorar o implementar sus propios códigos en el pipeline gráfico. Esto se hace a través de los llamados *Vertex Shader* y *Pixel/Fragment Shader*, figura III.7.

El *Vertex Shader* (figura III.8) es un código que suministra el programador, este shader tiene información del vértice (coordenada del vértice, normales, tangente, binormal, etc), el programador captura esta información y procede hacer algún tipo de cálculo con él. Lo mismo pasa con el *Pixel Shader*, es un código que tiene como entrada la información de un píxel (coordenada U y V, información de la textura, etc), este píxel es el resultado del proceso de rasterización de la tarjeta gráfica, y el programador en su código tiene que dar el color final del píxel.

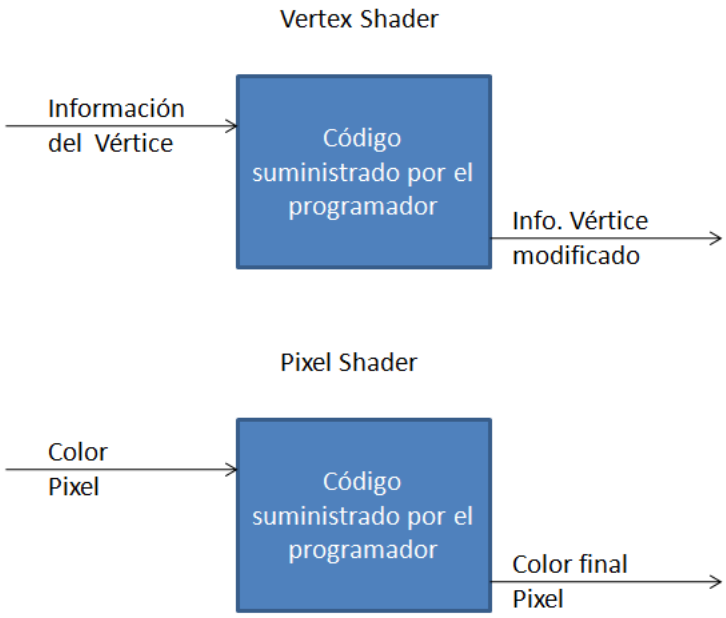


Figura III.8: *Funcionamiento del Vertex Shader y Pixel Shader*

Con esta nueva idea se abrieron un sin limites de efectos gráficos, algunos efectos como relieves (*bump mapping*, figura III.9), campo de profundidad (*depth of field*, figura), entre muchos otros, que eran tareas que se realizaban en el CPU, empiezan a calcularse usando el GPU, algo que abrió una generación de vídeo juegos con efectos espectaculares. En la página de desarrolladores de NVIDIA, se puede encontrar de manera gratuita 3 libros (GPU Gems) [28] que contienen una variedad de efectos visuales optimizados para el GPU usando Vertex Shader y Pixel Shader.

2.2 Paralelismo en GPU

Es importante destacar que el GPU no procesa un vértice o píxel a la vez. Los procesadores gráficos implementan el SIMD, es decir, el procesador toma un segmento de vértices o píxeles de la memoria, y les aplica operaciones simultáneas a dichos vértices o píxeles. Por esa razón, tarjetas gráficas como la Nvidia 6600 puede procesar 375 millones de vértices por segundo [31].

En la actualidad los GPU aprovechan ahora la tecnología multicore, pero debido a que el GPU es un procesador más simple que el CPU, se pueden introducir una cantidad considerables de núcleos (cores) en el GPU, como se muestra en el siguiente listado:

- Geforce 8800 GT - 112 cores.
- Geforce 9800 GTX - 128 cores.
- Geforce GTX 295 - 2 procesadores - 480 cores (240 por procesador).

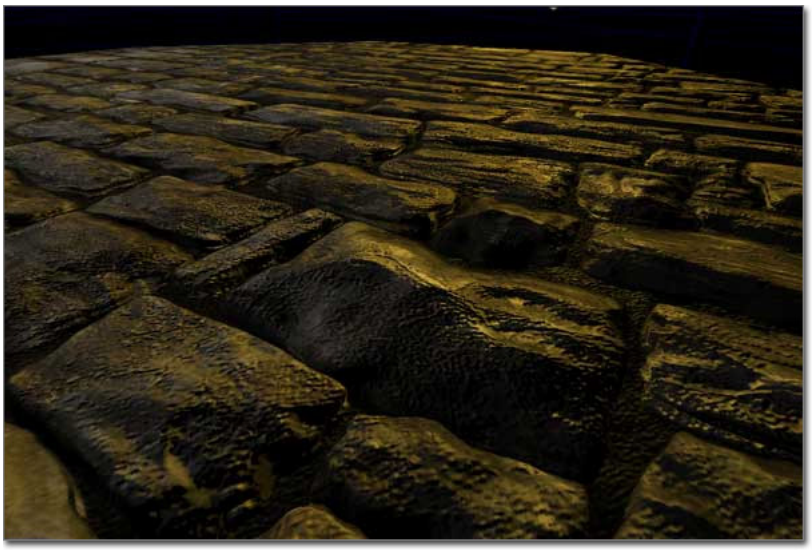


Figura III.9: *Relieve - Bump Mapping [29]*

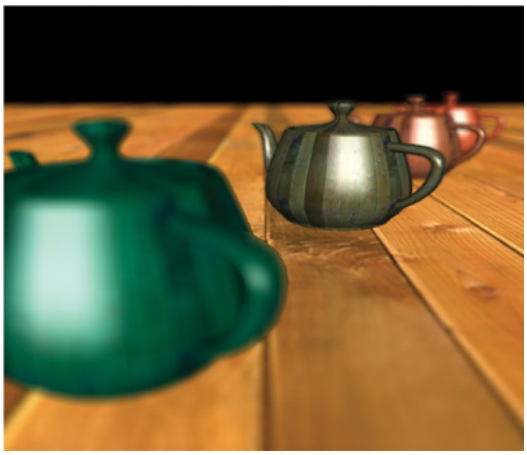


Figura III.10: *Campo Profundidad - Depth of Field [30]*

- Geforce GTX 460 - 336 cores.
- Geforce GTX 580 - 512 cores
- Quadro 600 - 448 cores.

Una de las diferencia tal vez entre el CPU y el GPU, es que el proceso paralelo es casi transparente para el diseñador. Es decir, si un programador levanta en la memoria n cantidad de vértices o n cantidad de píxeles, el programador no debe alterar el código, para que este funcione en forma paralela. Solo debe aplicar el *Vertex Shader* y el *Fragment Shader* correspondiente y el GPU automáticamente corre los códigos en paralelos sin que el usuario lo indique en su programa.

Esta simplicidad se debe, a que la tarjeta gráfica asume que todos los hilos que ejecuta (vértices o píxeles) son independientes, es decir, cuando el *vertex shader* evalua un vértice,

no se puede acceder a la información de otro vértices, igual sucede con los píxeles. Como se ve en el figura III.11, esta es una forma paralela de realizar cálculos, los programadores se dieron cuenta de esto y surge una comunidad (http://gpgpu.org/) *General-Purpose Computation on Graphics Hardware*, esta comunidad tiene como objetivo divulgar programas, papers, documentos o tutoriales, que usan el GPU para soluciones gráficas y no gráficas.

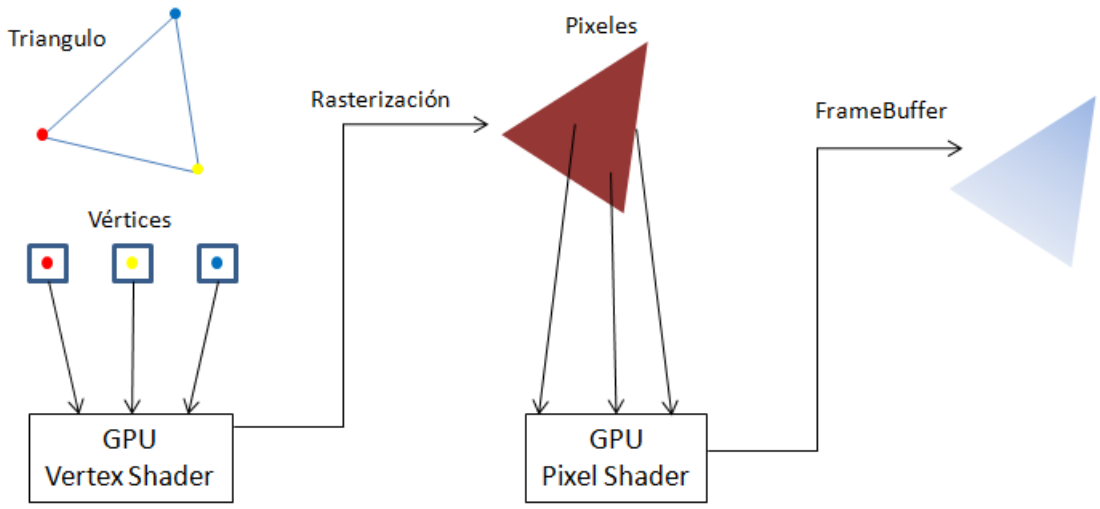


Figura III.11: *Paralelismo en GPU*

Es decir, los programadores empiezan a usar el GPU, no para un resultado final gráfico, sino para un resultado numérico. Por ejemplo, supongamos que se quiere sumar dos matrices 100x100. Los programadores se dieron cuenta que una matrix la podían almacenar en una textura en la memoria de la tarjeta gráfica, luego aplicando un *píxel shader*, podían tener acceso a las dos matrices, y crear una "imagen" que era la suma de las dos matrices, luego se captura el resultado en la memoria de la tarjeta gráfica. Como se puede ver en ningún momento se obtiene algún resultado gráfico, sino un resultado enteramente numérico.

No necesariamente computistas gráficos ahora usaban el GPU, sino científicos de otras disciplinas empiezan a usar este potencial para solucionar problemas no necesariamente gráficos:

- A Practical Quicksort Algorithm for Graphics Processors [32]
- A graphics hardware accelerated algorithm for nearest neighbor search [33]
- An Acceleration Technique for the Computation of Voronoi Diagrams Using Graphics Hardware [34]
- Fast collision detection using the A-buffer [35]

- Parallel processing of matrix multiplication in a CPU and GPU heterogeneous environment [36]
- Visualization and GPU-accelerated simulation of medical ultrasound from CT images [37]
- Accelerated 2D Image Processing on GPUs [38]

2.3 CUDA - Compute Unified Device Architecture

Hasta el inicio del 2007, si un programador deseaba usar el GPU para hacer algún cálculo tenía que aprender los fundamentos de la computación gráfica, además de conocer las librerías OpenGL o DirectX, esto era algo un poco tedioso para científicos que no estaban acostumbrado al área de gráficas, pero fue en el inicio del 2007 cuando NVIDIA da a conocer CUDA, para aquel entonces NVIDIA implementa CUDA en las tarjetas gráficas geforce 8800 GTX.

CUDA es un lenguaje basado en C, un código CUDA se compila para correr en el GPU, este lenguaje hace desaparecer toda librería gráfica (OpenGL o DirectX) para usar el poder del GPU, cualquier científico familiarizado con C podía programar en el GPU sin tener conocimiento ninguno en el área de gráficas. Inmediatamente empezaron a surgir aplicaciones, mejoras y optimizaciones usando este lenguaje. En la actualidad la página (http://www.nvidia.com/object/cuda_apps_flash_new.html) cuenta con cientos de aplicaciones, que van desde Imágenes medicas hasta Finanzas, es decir, empieza un auge por el uso del GPU para cualquier cálculo que se pueda correr en paralelo.

Teóricamente CUDA puede ser un poco intimidante, pero en la práctica su aplicación es mucho más simple. Antes de explicar como funciona CUDA se dan las siguientes definiciones, (figura III.12):

- *Kernel* (función): es una función de C, que reside en el GPU y es la que se ejecuta en forma paralela.
- *Thread* (hilo): un hilo es quien ejecuta un kernel.
- *Block* (bloque): es un conjunto de hilos. Es decir, un bloque puede tener hasta 512 hilos, estos hilos pueden estar en un bloque de manera unidimensional, bidimensional o tridimensional.
- *Grid*: es un conjunto de bloques, estos bloques pueden estar en forma unidimensional o bidimensional.

CUDA a su vez tiene diferentes tipos de memorias (figura III.13) para almacenar la información, estas son:

- *Register* (registro): es un espacio de memoria para cada hilo, esta memoria reside en el chip del núcleo.

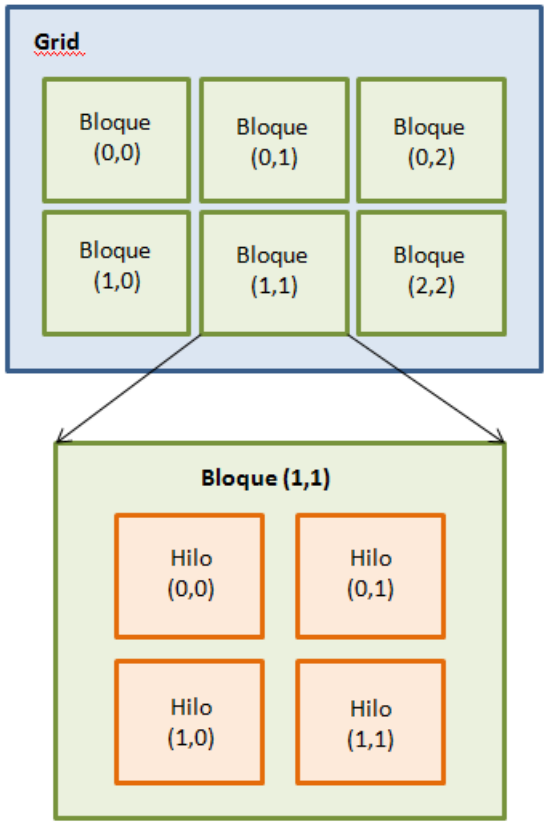


Figura III.12: *CUDA: Grid, Bloque e Hilo*

- *Shared Memory* (memoria compartida): es una memoria que comparten todos los hilos de un bloque, y también reside en el chip del núcleo.
- *Global Memory* (memoria global): es la memoria de la tarjeta gráfica, a esta memoria tienen acceso todos los bloques.
- *Texture Memory* (memoria textura): también reside en la memoria de la tarjeta, a esta memoria tienen acceso todos los bloques. Se diferencia de la memoria global ya que se accede de manera diferente y se le aplican algunas operaciones automáticamente, como por ejemplo, algunos algoritmos de filtro.
- Constant Memory (memoria constante): es una memoria para almacenar solo variables en modo lectura, a esta memoria tienen acceso todos los bloques.

Una vez definidos estos términos, se explica ahora el funcionamiento de CUDA, en primera instancia, las tarjeta de vídeo NVIDIA tienen Multiprocesadores (*Streaming Multiprocessor SMs*), cuando una aplicación corre un kernel (función), la ejecución de este kernel se realiza a través de un grid, dicho grid se conforma por bloques y cada bloque tienen los hilos que ejecutarán al kernel. Un bloque se ejecuta en un solo multiprocesador.

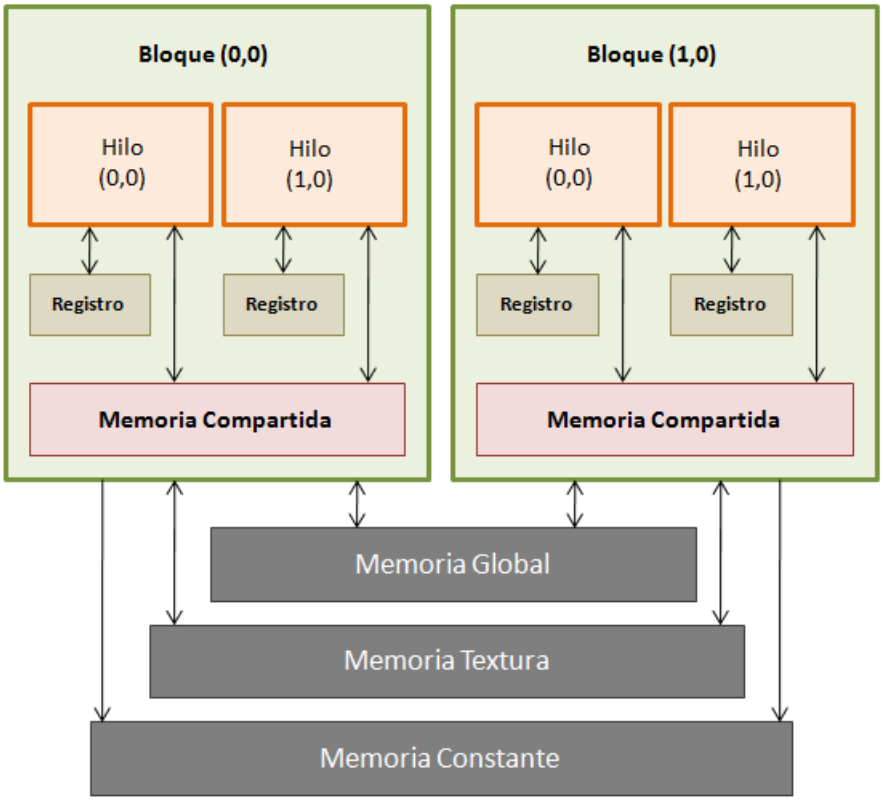


Figura III.13: *Memoria en CUDA*

Adicionalmente cada multiprocesador posee 8 procesadores escalares (Scalar Processor SP), NVIDIA suele usar el nombre **CUDA cores** para estos procesadores, estos SP son los encargados de crear, organizar, y ejecutar los hilos en forma paralela. Es decir, el SM ordena la ejecución de un hilo a travez de un SP. A esta arquitectura NVIDIA le da el nombre de SIMT (single-instruction multiple-thread) que es una extensión del SIMD.

Un SM puede ejecutar más de un bloque al mismo tiempo (la arquitectura de hardware actual permite ejecutar hasta 8 bloques simultáneamente), para mantener cierto orden en la ejecución, el SM, crea, maneja y ejecuta los hilos en forma paralela en grupo de 32, a dicho grupo se le da el nombre de *warp*. Luego cada SP ejecuta un hilo determinado de ese grupo.

Cabe mencionar que no existe un orden de ejecución en los bloques, ni un orden de ejecución en los hilos. Es decir, si hay 10 bloques enumerados del 0 al 9, el GPU puede ejecutar el 9 primero y el 0 de último, a su vez cuando un bloque entra en ejecución en un SM, y este reparte los hilos a los SP, tampoco hay un orden para su ejecución, puede suceder que el último hilo del bloque se ejecute de primero, y el primer hilo del bloque de último. La única diferencia, en el caso de los bloques, es que los hilos pueden sincronizar su ejecución, esto se usa cuando los hilos comparten información a través de la memoria compartida (*shared memory*). Pero no existe comunicación entre los bloques.

También cada tarjeta gráfica posee algunas limitaciones, es decir, la Geforce 9600 GT tiene las siguientes características:

- Multiprocesadores: 8
- CUDA core o SP: 8*8 = 64
- Total de memoria constante = 65536 bytes
- Total de memoria compartida por bloque: 16384 bytes
- Total de numero de registros por bloque: 8192
- Tamaño del *warp*: 32
- Máximo de hilos por bloque: 512
- Máxima dimensión de un bloque: 512x512x64 hilos
- Máxima dimensiona de un *grid*: 65535x65535x1 bloques

Estos valores pueden variar un poco dependiendo del modelo de la tarjeta gráfica, ahora, estos valores se pueden capturar en la ejecución de un programa, por consiguiente, cualquier programa se puede adaptar a cualquier tarjeta de vídeo. El programador será el encargado de crear un código que se adapte a estas limitaciones, en caso de violar algunas de ellas, como por ejemplo, usar más memoria compartida de lo disponible, el *kernel* no se ejecutará y retornará un error.

Se muestra a continuación un sencillo ejemplo, que consta en multiplicar un escalar por un vector.

Algoritmo III.1: Código Cuda Multiplicacion vector-escalar

```
1
2  __global__ void MultiplicarVector(float *arr, int N, float esc)
3  {
4    int idx = blockIdx.x;
5    arr[idx] = esc * arr[idx];
6  }
7
8  int main(void)
9  {
10   N =100;
11   int *a_d;
12
13   ...
14
15   MultiplicarVector <<< N, 1 >>> (a_d, N, 3);
16
17   ...
18 }
```

En primera instancia se tiene el *kernel* (MultiplicarVector), para que el GPU reconozca que esta función es un kernel hay dos reglas, la primera es, ser una función que no retorne ningún valor (void) y segundo debe ser declarada con el argumento _ _ global _ _. A esta función se le pasa el vector que se quiere alterar (arr), la cantidad de elementos del vector (N) y el escalar que se quiere multiplicar por el vector (esc).

Como segundo punto importante, tenemos la invocación de la función, que se diferencia un poco del estándar de C, como se ve en la línea 16 se tiene los símbolos <<< >>>, dentro de estos símbolos, se tienen que especificar la cantidad de bloques que se desea crear, y la cantidad hilos que contiene cada bloque respectivamente, en este caso se están creando 100 bloques de un hilo cada uno.

Lo otro importante a señalar es lo siguiente, como se mencionó mas arriba, el GPU puede ejecutar los bloques en cualquier orden, es decir, no es secuencial, por esa razón la línea de código 15, *blockIdx.x* retorna el ID del bloque que se esta ejecutando, este número va del 0 al 99, misma dimensión que el arreglo *arr*.

Como se aprecia en el código, el programador es el encargado de proporcionar la cantidad de bloques e hilos de cada bloque, siempre y cuando dichos valores esten dentro de las limitaciones del hardware como se presentó en la parte arriba. Por ejemplo en el caso de que $N = 1000000$, ya el código tendría que cambiar a lo siguiente:

Algoritmo III.2: Código Cuda Multiplicacion vector-escalar

```
1
2  __global__ void MultiplicarVector(float *arr, int N, float esc)
3  {
4    int idx = blockIdx.x * blockDim.x + threadIdx.x;
5
6    if (idx<N) a[idx] = escalar * a[idx];
7  }
8
9  int main(void)
10 {
11   N =1000000;
12   int *a_d;
13
14   ...
15   int block_size = 256;
16   int n_blocks = N/block_size + (N % block_size == 0 ? 0:1);
17
18   MultiplicarVector <<< n_blocks, block_size >>> (a_d, N, 3);
19
20   ...
21 }
```

En este caso, se fija cuantos hilos tendrán cada bloque (línea 15), luego se calcula cuantos bloques se necesitan para cubrir el tamaño de N (línea 16). La otra variación se encuentra en la línea 4, esa fórmula permite calcular el índice del arreglo al cual se esta accediendo en la función.

La decisión de cuantos hilos o bloques se necesitan para solucionar un problema dependerá de varias variables, en el capítulo 4 se específica un poco más este tema, aunque se puede encontrar información detallada y específica en las referencias [39] [40].

2.4 CPU vs GPU

En la parte superior, se muestra que el GPU puede tener más núcleos que el CPU, pero aún con eso, existe la interrogante del por qué el GPU puede acelerar algunos cálculos hasta 100x con respecto al CPU. Para responder esta interrogante se necesitan dos definiciones:

- *Flops* (figura III.14): es el acrónimo de *Floating point Operations Per Second* (operaciones de punto flotante por segundo). Y se usa para medir el rendimiento de un computador, o en este caso de los procesadores, en la actualidad los CPU y GPU se miden en Gigaflops.
- *BandWidth Memory* (figura III.15): es la velocidad con que la información se lee o escribe en la memoria. En la actualidad los CPU y GPU se miden en *GigaBytes* por segundo.

En la figura III.14, se puede ver claramente como el GPU es muy superior en término de *Flops*, mas aún, el último procesador Intel i7 alcanza los 42.56 *GigaFlops*/seg [41], muy por debajo de los casí 1000 *GigaFlops*/seg, de las tarjetas NVIDIA serie GT200.

En el caso del ancho de banda (*BandWidth*) existe una enorme diferencia entre la arquitectura Intel, y los procesadores NVIDIA, estas dos razones son la clave importante del por qué en la actualidad los software están empezando a optimizar sus programas usando el GPU.

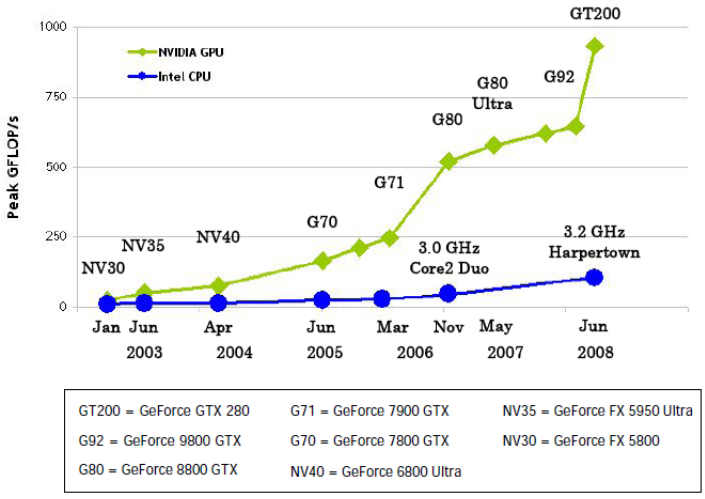


Figura III.14: *GigaFlops del CPU y GPU [42]*

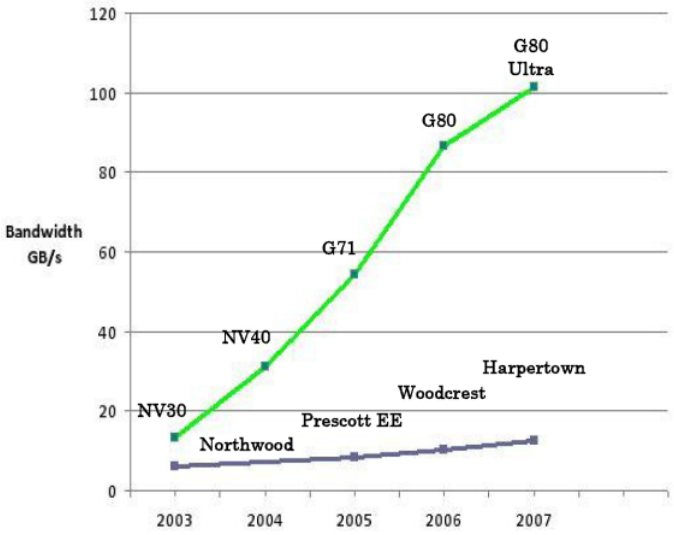


Figura III.15: *BandWidth/seg del CPU y GPU [42]*

3 Intersección y Colisiones

La detección de colisiones se utiliza en una gran variedad de aplicaciones, que incluye vídeo juegos, simulación física (animación por computadora), robótica, ingeniería de simulación, etc. Colisionar, básicamente consiste en indicar si dos objetos se están intersectando. Un objeto es una estructura que puede representarse de varias maneras, por ejemplo, puede ser una malla geométrica formada por una secuencia de triángulos o podría ser una expresión algebraica como por ejemplo, una esfera, cilindro, NURBS, etc.

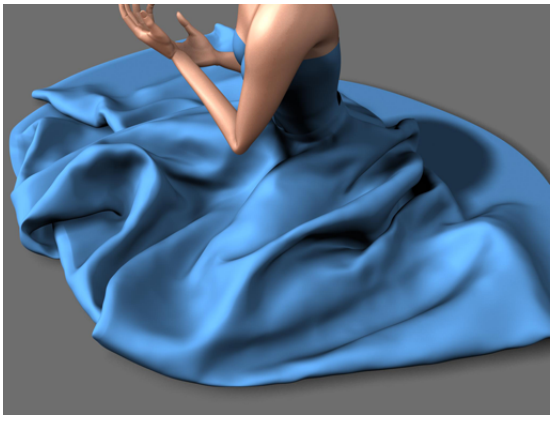


Figura III.16: *Colisiones - Malla deformable [43]*

Dependiendo de como sea el objeto a intersectar se pudieran aplicar distintos algoritmos o técnicas de colisión. En este trabajo se utilizan mallas geometrías formada por triángulos. En las referencias [44] y [45] pueden encontrar otro tipos de estructuras.

El estudio de las colisiones en mallas poligonales se dividen en dos, la primera se refiere a la colisión de objetos rígidos (rigid bodies), es decir, objetos que no se deforman [46], muy utilizado en el mundo de los vídeos juegos, y el segundo se refiere a colisión de objetos suaves o deformables (soft body), esta colisión es más compleja que la primera, ya que el objeto se puede deformar en el transcurso de una animación, la técnica que se usa para simular la tela o ropa (cloth, figura 3.16) [47] es un ejemplo de una superficie deformable que se estudia con mucho énfasis en la computación gráfica.

Cuando se utilizan objetos rígidos, usualmente no es común hacer una intersección triángulo a triángulo de cada objeto (a menos que el problema lo requiera), las técnicas que se emplean consiste en encerrar los objetos a colisionar en mallas poligonales más simples y se procede a colisionar dichas mallas, a estas se les conoce como *objeto de borde* (Bounding Volume BV). A continuación se muestran los distintos BV que se utilizan [48]:

- **Caja de borde alineada - *Axis-aligned Bounding Boxes* (AABB)**: este BV es uno de los más simples (figura III.17), y consiste en encerrar la geometría a colisionar en una caja, los ejes de esta caja están alineados a los ejes del objeto en cuestión, y además es la mínima caja que contiene a dicho objeto.

Figura III.17: *Caja de borde alineada - AABB*

  La colisión de los AABB es muy rápida, pero no son muy preciso, ya que puede suceder que dos objetos no colisionen, pero sus AABB si pudieran colisionar, no obstante esta técnica sigue siendo utilizada en los vídeos juegos y en la animación por computadora.

- **Esferas - *Spheres***: Esta se podría decir que es la más simple de todas (figura III.18), pero a su vez la más imprecisa, consiste en encerrar la malla geométrica en una esfera, y detectar la colisión de los objetos, colisionando las esferas, algo que es extremadamente rápido.

- **Caja Orientada - *Oriented Bounding Boxes* (OBB)**: Similar al AABB, con la diferencia que aquí la caja esta orientada de una manera que el objeto encaje con mayor precisión en su interior y así mejorar la posibilidad de que exista alguna colisión entre los objetos (figura III.19).

Figura III.18: *Esfera*

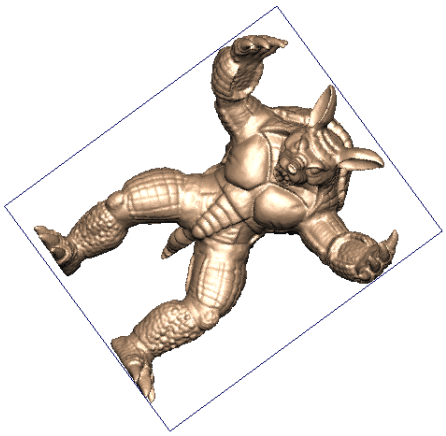


Figura III.19: *Caja Orientada - OBB*

Este algoritmo, tiene el problema en determinar los ejes más óptimos para crear la caja que encierre al objeto. En la referencia [49] se encuentra una técnica muy óptima para encontrar dichos ejes.

- **Cápsula Convexa - *Convex Hull***: Con este método, se encierra el objeto a colisionar en uno o varios polígonos convexos (figura III.20), aunque esta técnica es una de las más lenta, da una mayor precisión para determinar la colisión de varios objetos.

  En la figura III.20 a la izquierda se puede apreciar un objeto encerrado en dos cápsulas convexas, y en la derecha el mismo objeto encerreda en 4 cápsulas convexas.

Estos BV algunos más rápidos que otros en objetos rígidos solo se calcula una sola vez, ya que los objetos no cambian su forma, es por esa razón que los objetos deformables trabajan con otras técnicas, eso no quiere decir que no se puedan aplicar estos BV a objetos deformables como la referencia [51] que aplica AABB a objetos que se deforman, muy interesante esta técnica, ya que no crea un AABB para todo el objeto, sino que crea un arreglo de AABB que encierra al objeto deformable.

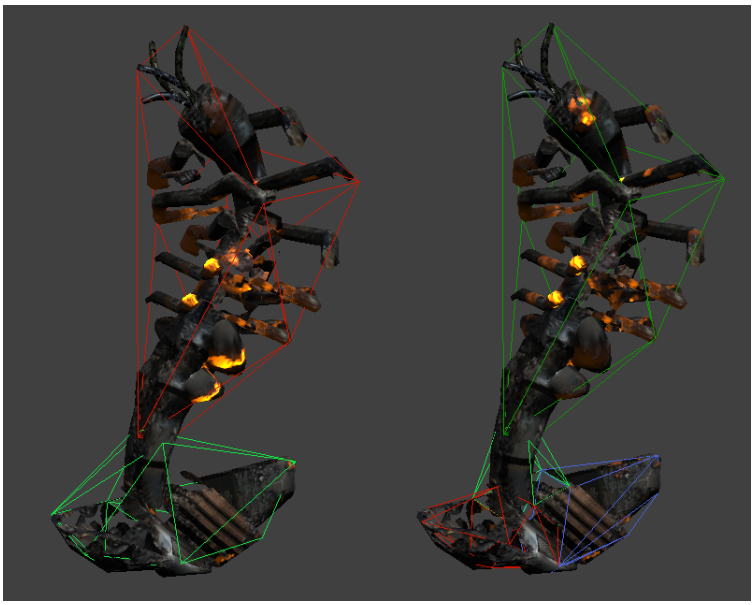


Figura III.20: *Cápsula Convexa, Unreal Development Kit [50]*

No obstante, en algunas ocasiones interesa intersectar los dos objetos y conocer los triángulos de intersección, sin embargo, colisionar cada triangulo de un objeto contra todos los triángulos del otro objeto, no sería la manera más eficiente de hacerlo, es por ello que se considera la partición del espacio en algunas estructura y así optimizar el cálculo.

Existen muchas estructuras que se pueden utilizar como OBBTree [52], BOXTREE [53], R-tree [54], Quadtree/Octree [55] y BSP tree [56] [48], este último es uno de los utilizado en la actualidad.

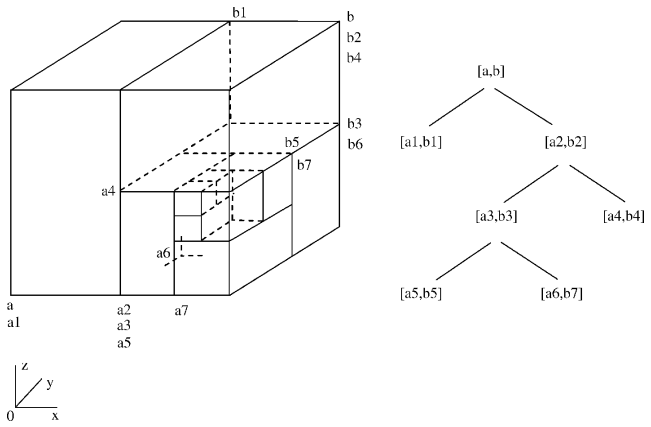


Figura III.21: *Binary space-partitioning*

Básicamente el BSP (*Binary space-partitioning*, figura III.21) es un proceso recursivo que divide la escena en dos hasta que la partición satisface cierta condición.

En término de colisión, por ejemplo, intersectando un segmento de recta con un objeto poligonal, se aplicaría lo siguiente:

- Se crea el BSP del objeto (por ejemplo, se puede tomar como condición que cada caja del BSP contenga 100 triángulos)
- Se intersecta el segmento de recta con cada caja del BSP
- Se toma cada caja que el segmento intersecta y se procede a intersectar la recta con cada triángulo.

Así se optimizaría la intersección de un segmento de recta con un malla poligonal. Un resumen de las técnicas de colisión pueden ser encontrados en las referencias [44] y [45].

CAPÍTULO IV

IMPLEMENTACIÓN

En los próximos dos capítulos se muestran los resultados obtenidos al implementar las intersecciones usando la geometría conforme, se desea comparar el tiempo de ejecución de los algoritmos usando el CPU y el GPU, pero antes de especificar con más detalle los resultados, se presentan los siguientes puntos:

- Para realizar cualquier tipo de intersección no se empleo ningún BV (*Bounding Volume*) o alguna estructura como BSP vistas en el capítulo 3, la idea es ver como se comportan en tiempo de cómputo las intersecciones en el modelo conforme usando el peor de los casos, que es haciendo la verificación triángulo a triángulo.
- Los modelos utilizados para este trabajo fueron tomados del repositorio de la Universidad de Stanford. (http://www.graphics.stanford.edu/data/3Dscanrep/).

1 Algoritmos

En el capítulo 2 se presentó las condiciones para que las intersecciones entre los distintos objetos del modelo conforme se dieran, pero no basta con solo esas condiciones, por ejemplo, no es de interés intersectar dos rectas infinitas, se quiere intersectar dos segmentos de rectas, así mismo en vez de un plano infinito se desea intersectar un triángulo con los otros objetos. A continuación se presentan los algoritmos utilizados para cada intersección.

1.1 Intersección Segmento de Recta-Segmento de Recta

Pseudocódigo:

Algoritmo IV.1: Intersección segmento-segmento

```
1
2  Procedimiento InterseccionSegmentoSegmento(R1, R2 )
3
4    //Interseccion de los segmentos R1 y R2
5     Normalizar(R1)
6     Normalizar(R2)
7
8     a = CalculoInterseccionModeloConformeRectaRecta(R1,R2)
9
10    Si a es verdadero entonces
11
12       rvecs1_1 = R2.punto1 - R1.punto1
13       rvecs1_2 = R2.punto2 - R1.punto1
14       rvecs1_3 = R1.punto2 - R1.punto1
15
```

```
16        rvecs2_1 = R1.punto1 - R2.punto1
17        rvecs2_2 = R1.punto2 - R2.punto1
18        rvecs2_3 = R2.punto2 - R2.punto1
19
20        signo1 = bivector(rvecs1_3,rvecs1_1)
21        signo2 = bivector(rvecs1_3,rvecs1_2)
22
23        signo3 = bivector(rvecs2_3,rvecs2_1)
24        signo4 = bivector(rvecs2_3,rvecs2_2)
25
26        //Se verifica los signos de los bivectores
27        Si signo1 = signo2 entonces
28             Imprimir "No hay Intersección de los dos segmentos"
29        si no entonces
30              Si signo3 = signo4 entonces
31                 Imprimir "No hay Intersección de los dos segmentos"
32              si no entonces
33                 Imprimir "Si hay intersección de los dos segmentos"
34              Fin si
35        Fin si
36    Fin si
37
38 Fin procedimiento
```

Se empieza normalizando las rectas (en la sección de implementación se explica la razón de esta normalización). Luego se hace el cálculo de intersección en el modelo conforme usando las fórmulas y condiciones dadas en sección (2.3.1), pero en el caso que exista intersección se verifica si los dos segmentos se intersectan.

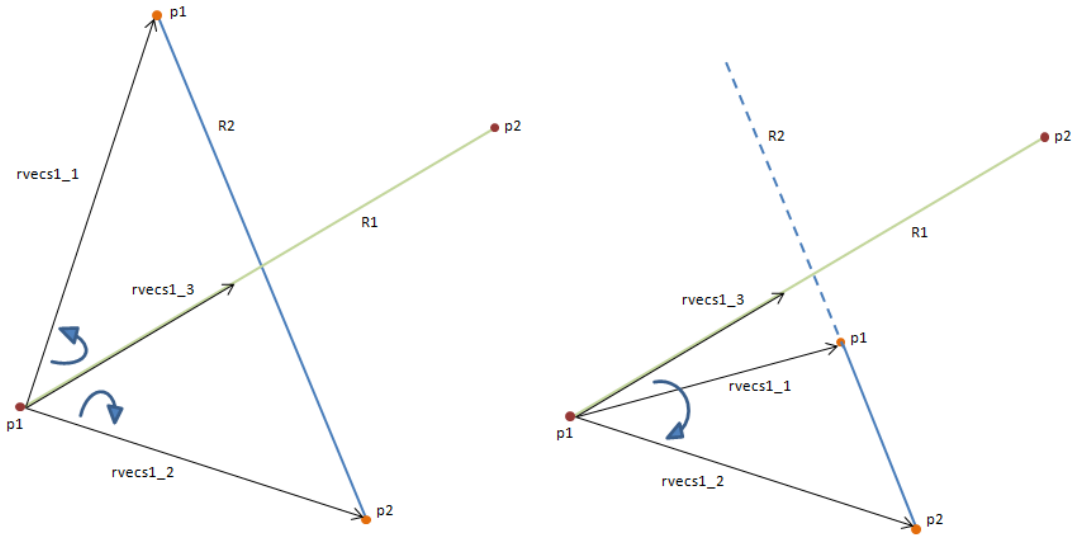


Figura IV.1: *Izquierda: se intersectan los segmentos (signos diferentes), Derecha: No se intersectan (signos iguales)*

Para verificar si los dos segmentos se intersectan, se calcula la orientación de los bivectores $rvecs1\_3 \wedge rvecs1\_2$ y $rvecs1\_3 \wedge rvecs1\_1$ definidos como se muestra en la figura IV.1, si tienen orientación opuesta (signos diferentes) entonces los segmentos se intersectan, si poseen la misma orientación (signos iguales) los dos puntos están del mismo lado, por lo tanto, los segmentos no se intersectan.

1.2 Intersección Segmento de Recta-Triángulo

Pseudocódigo:

Algoritmo IV.2: Intersección segmento-triángulo

```
1
2  Procedimiento InterseccionSegmentoPlano(R1, P1 )
3
4    //Interseccion del segmento R1 con el triángulo (plano) P1
5     Normalizar(R1)
6     Normalizar(P1)
7
8    [a,e3er] = CalculoInterseccionModeloConformeRectaPlano(R1,R2)
9
10   R2 = Recta(P1.punto3,P1.punto1)
11   R3 = Recta(R1.punto1,R1.punto2)
12   [out1,ind1] = InterseccionSegmentoSegmento(R2, R3 )
13
14   R2 = Recta(P1.punto1,P1.punto2)
15   R3 = Recta(R1.punto1,R1.punto2)
16   [out2,ind2] = InterseccionSegmentoSegmento(R2, R3 )
17
18   R2 = Recta(P1.punto3,P1.punto2)
19   R3 = Recta(R1.punto1,R1.punto2)
20   [out3,ind3] = InterseccionSegmentoSegmento(R2, R3 )
21
22    // out# variables que determina si los segmentos se intersectan: 1 (
         intersecta) 0 (no hay intersección o reside en el plano)
23    // ind# escalar que multiplica al vector e
24
25    Si a = 0 entonces
26          //La recta y el plano son paralelos
27
28       Si e3er = 0 entonces //La reta reside en el plano
29
30          //Se verifica la intersección del segmento con los segmentos que
               forman al triángulo
31          Si out1=1 o out2=1 o out3=1 entonces
32             Imprimir "Si hay interseccion"
33          si no entonces
34             Imprimir "No hay interseccion"
35          Fin si
36
37       si no entonces
38          Imprimir "No hay intersección"
39       Fin si
```

```
40
41   si no entonces
42
43      //Se determina si los dos puntos están de un lado del plano
44
45       signo1 = trivector(P1.punto2-P1.punto1,P1.punto3-P1.punto1,R1.punto1-
             P1.punto1)
46       signo2 = trivector(P1.punto2-P1.punto1,P1.punto3-P1.punto1,R1.punto2-
             P1.punto1)
47
48       Si signo1 = signo2 entonces
49          Imprimir "No hay intersección"
50          Salir del Procedimiento
51       Fin si
52
53
54       // La recta atraviesa el plano
55       salida = 0
56       Si ind1>0 y ind2 >0 y ind3<0 entonces
57          salida = 1
58       Fin si
59
60       Si ind1<0 y ind2 <0 y ind3>0 entonces
61          salida=1;
62       Fin si
63
64
65       Si salida = 0 entonces
66          Imprimir "No hay intersección"
67       si no entonces
68          Imprimir "Si hay intersección"
69       Fin si
70
71   Fin si
72 Fin procedimiento
```

Para intersectar una recta con un plano hay que considerar lo siguiente, si la recta es paralela al plano hay dos posibilidades, o la recta vive en el plano o reside fuera de él. El escalar $e3e = (-\omega_3\beta_3 + \omega_4\beta_2 + \omega_1\beta_6)$ determina si la recta está o no en el plano. Pero se quiere intersectar un segmento con un triángulo, por lo tanto, si la recta vive en el plano, se toma el segmento de recta y se intersecta con los segmentos de rectas que forman al triángulo.

Por otro lado, si la recta intersecta al plano, no necesariamente intersecta al triángulo (figura IV.2), para determinar si la recta intersecta al triángulo, se verifica un escalar que se toma de la intersección de la recta con las tres rectas del triángulo, este escalar es $(\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1)$ que se toma de la ecuación (A.2.1), el signo de este escalar ayuda a determinar si la recta intersecta o no al triángulo, como se ve en el algoritmo en las líneas 65 y 69.

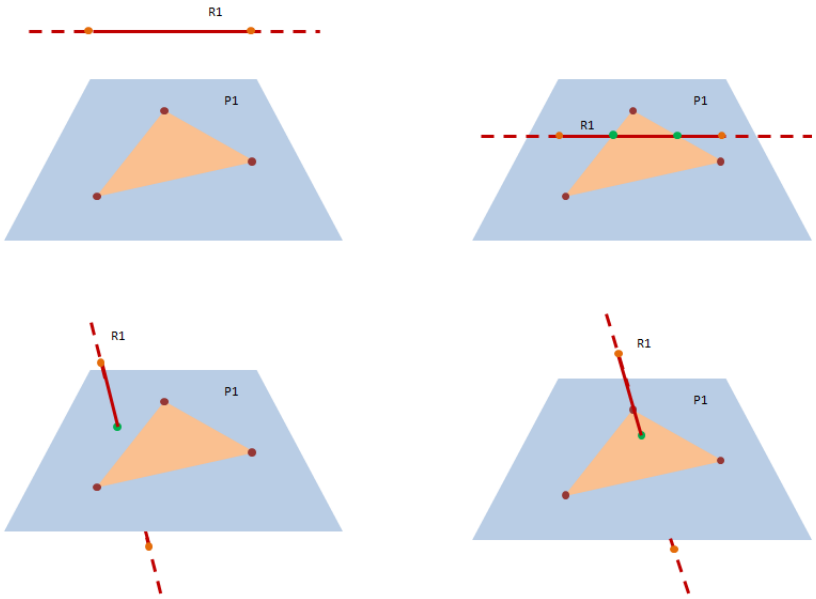


Figura IV.2: *Superior-Izquierda: e3e = 0 la recta es paralela pero no reside en el plano, Superior-Derecha: e3e ≠ 0 la recta es paralela pero vive en el plano. Inferior-Izquierda: la recta no intersecta al triángulo, Inferior-Derecha: La recta intersecta al triángulo*

1.3 Intersección Segmento de Recta-Esfera

Pseudocódigo:

Algoritmo IV.3: Intersección segmento-esfera

```
1
2  Procedimiento InterseccionSegmentoEsfera(E1, R1 )
3
4    //Intersección del segmento R1 con la esfera E1
5
6    //Se coloca el origen en el centro de la esfera
7    CambioCoordenadasEsfera(E1)
8
9    //Se reescriben los puntos de la recta en el nuevo sistema
10   CambioCoordenadasRecta(R1)
11
12   //Se verifica si uno de los puntos esta en el interior de la esfera
13   b = VerificarPuntosInteriorEsfera(R1)
14
15   Si b = verdadero entonces
16      Imprimir "EL segmento de recta intersecta la esfera"
17      Salir procedimiento
18   Fin si
19
20   Normalizar(R1)
21
22   a = CalculoInterseccionModeloConformeRectaEsfera(R1,E1)
23
```

```
24     Si a <= 0 entonces
25         Si a = 0 entonces
26             Imprimir "El segmento de recta es tangente a la esfera"
27         si no entonces
28             Imprimir "El segmento de recta no intersecta la esfera"
29         Fin si
30
31   si no entonces
32
33       vecOP1 = -R1.punto1
34       vecOP2 = -R1.punto2
35
36       vecP1P2 = R1.punto2 - R1.punto1
37       vecP2P1 = R1.punto1 - R1.punto2
38
39       cos1 = CosenodelAngulo(vecOP1,vecP1P2)
40       cos2 = CosenodelAngulo(vecOP2,vecP2P1)
41
42       // Si ambos ángulos son agudos el segmento intersecta la esfera, si
             uno es obtuso los dos puntos del segmento están afuera de la
             esfera
43       Si cos1 < 0 y cos2 < 0 entonces
44           Imprimir "El segmento de recta no intersecta la esfera"
45       si no entonces
46           Imprimir "El segmento de recta intersecta la esfera"
47       Fin si
48
49   Fin si
50
51 Fin procedimiento
```

En primera instancia para simplificar los cálculos se traslada el origen al centro de la esfera, luego se verifica si algún punto del segmento de recta se encuentra en el interior de la esfera, de ser afirmativo, implica que hay intersección, si los dos puntos se encuentran en el interior entonces no hay intersección.

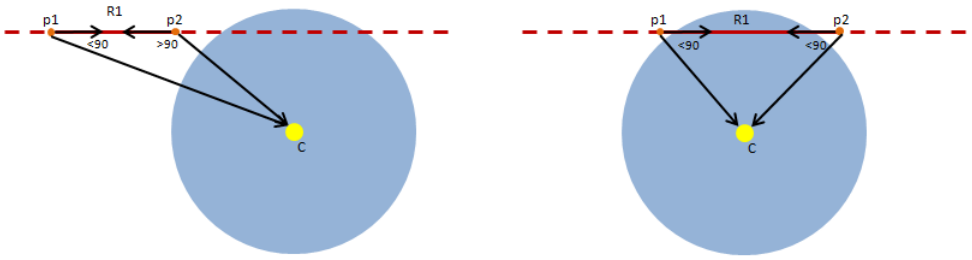


Figura IV.3: *Izquierdo: uno de los angulos entre los vectores es >90. Derecha: ambos angulos son agudos*

Sin embargo, puede ocurrir que exista intersección de la recta con la esfera, pero los puntos del segmento de recta se encuentren en la parte externa de la misma. Para determinar este caso, se toman los ángulos de los vectores señalados en la figura IV.3, si uno de

los ángulos es obtuso entonces el segmento de recta se encuentra en el exterior de la esfera, y en caso de que ambos ángulos sean agudos, entonces el segmento de recta atraviesa la esfera.

1.4 Intersección Triángulo-Triángulo

Pseudocódigo:

Algoritmo IV.4: Intersección triángulo-triángulo

```
1
2  Procedimiento InterseccionSegmentoPlano(P1, P2 )
3
4    //Intersección del triángulo P1 con el triángulo P2
5
6     //Se verifica si los puntos de uno de los planos están de un solo lado
7    r =  VerificaLadoPuntos(P1)
8
9    Si r es verdadero entonces
10      Imprimir "No hay Intersección"
11      Salir Procedimiento
12   Fin si
13
14    Normalizar(P1)
15    Normalizar(P2)
16
17    r1 = Recta(P2.punto1,P2.punto2)
18    r2 = Recta(P2.punto1,P2.punto3)
19    r3 = Recta(P2.punto2,P2.punto3)
20
21    // out# = 0 no hay intersección, 1 si hay intersección
22   out1 = InterseccionSegmentoPlano(r1, P1 )
23   out2 = InterseccionSegmentoPlano(r2, P1 )
24   out3 = InterseccionSegmentoPlano(r3, P1 )
25
26   Si out1 = 1 o out2 = 1 o out3 = 1 entonces
27      Imprimir "Si hay interseccion"
28      Salir Procedimiento
29   Fin si
30
31   r1 = Recta(P1.punto1,P1.punto2)
32   r2 = Recta(P1.punto1,P1.punto3)
33   r3 = Recta(P1.punto2,P1.punto3)
34
35    // out# = 0 no hay intersección, 1 si hay intersección
36   out1 = InterseccionSegmentoPlano(r1, P2 )
37   out2 = InterseccionSegmentoPlano(r2, P2 )
38   out3 = InterseccionSegmentoPlano(r3, P2 )
39
40   Si out1 = 1 o out2 = 1 o out3 = 1 entonces
41      Imprimir "Si hay intersección"
42   si no entonces
43      Imprimir "No hay intersección"
44   Fin si
```

```
45
46
47 Fin procedimiento
```

En el caso de la intersección de dos triángulos no se utiliza la fórmula para intersectar dos planos en el modelo conforme, la razón se debe a que sin importar el resultado de la intersección en el modelo conforme había que aplicar la técnica que se explica a continuación, esta técnica ya determinaba la existencia o no de una intersección. El algoritmo consiste en tomar cada segmento que forma un triángulo e intersectarlo con el otro triángulo.

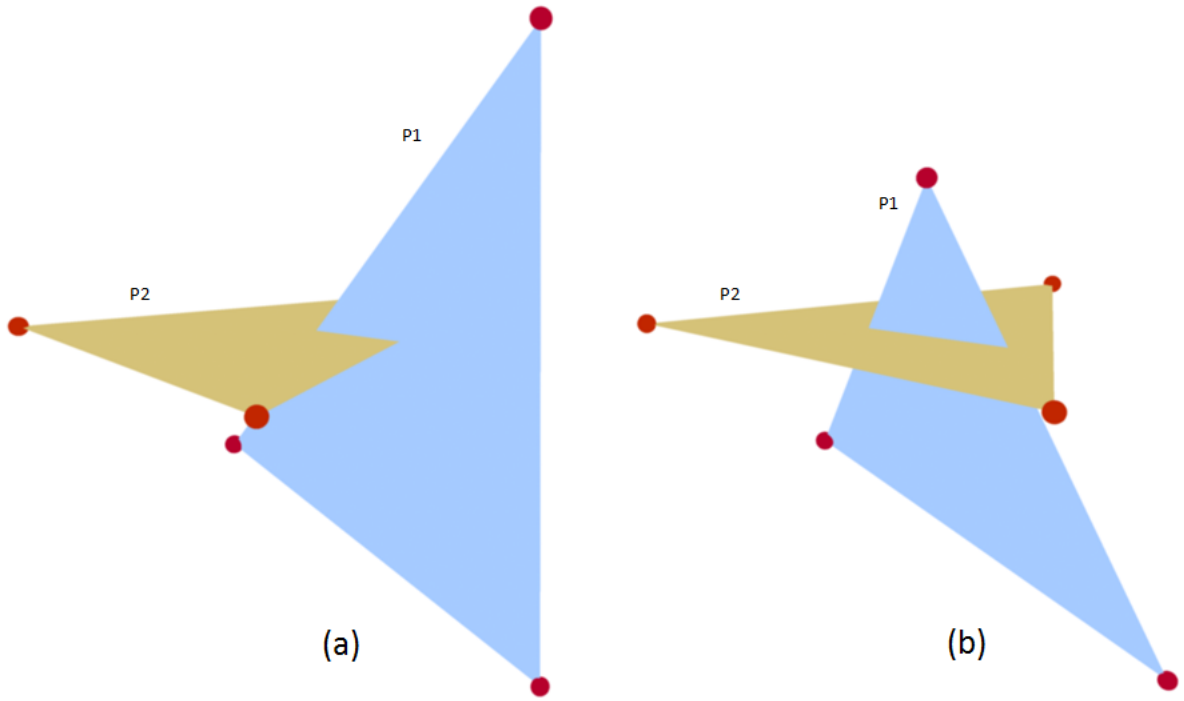


Figura IV.4: *Intersección Triángulo-Triángulo*

Si algún segmento intersecta al triángulo, entonces implica que los dos triángulos se intersectan, sin embargo, existe la posibilidad que en uno de los triángulos sus segmentos no intersecten al plano, y aun así exista intersección (Figura IV.4 (b)). Por esa razón, se toman los segmentos de ambos triángulos y se intersectan con cada triángulo. Esta intersección se realiza con el algoritmo 4.3.

1.5 Intersección Triángulo-Esfera

Pseudocódigo:

Algoritmo IV.5: Intersección triángulo-triángulo

```
1
2 Procedimiento InterseccionTrianguloEsfera(P1, E1 )
3
4   //Intersección del triángulo P1 con la esfera E1
```

```
5
6    //Se coloca el origen en el centro de la esfera
7   CambioCoordenadasEsfera(E1)
8
9   //Se reescriben los puntos de la recta en el nuevo sistema
10  CambioCoordenadasPlano(P1)
11
12   //Se verifica si algún punto se encuentra en el interior de la esfera,
         solamente no va ver intersección cuando los 3 puntos estén en el
         interior de la esfera.
13  r =  VerificaPuntosDentroEsfera(P1)
14
15  Si r es verdadero entonces
16     Imprimir "Si hay Intersección"
17     Salir Procedimiento
18  Fin si
19
20   Normalizar(P1)
21
22   r = CalculoInterseccionModeloConformePlanoEsfera(P1,E1)
23
24   Si r < 0 entonces
25     Imprimir "No hay intersección"
26  si no entonces
27
28     // Se verifica si una de las rectas del plano intersecta la esfera
29     r1 = Recta(P1.punto2,P1.punto1)
30     r2 = Recta(P1.punto3,P1.punto2)
31     r3 = Recta(P1.punto1,P1.punto3)
32
33     res1 = InterseccionSegmentoEsfera(E1,r1); // 0 = no intersecta, 1 =
          intersecta
34     res2 = InterseccionSegmentoEsfera(E1,r2);
35     res3 = InterseccionSegmentoEsfera(E1,r3);
36
37     Si res1 = 1 o res2 = 1 o res3 = 1 entonces
38        Imprimir "Si hay intersección"
39     si no entonces
40
41        //Se calcula el punto del plano cuya distancia sea mas corta al
             centro de la esfera.
42        [d, q] = CalculoPuntoMasCorto(E1,P1) // d = distancia, q punto en el
              plano
43
44        Si d <= r entonces
45
46            v12 = P1.punto2 - P1.punto1
47            v10 = q - P1.punto1
48
49            v23 = P1.punto3 - P1.punto2
50            v20 = q - P1.punto2
51
52            v31 = P1.punto1 - P1.punto3
53            v30 = q - P1.punto3
```

```
54
55              signo1 = bivector(v12,v10)
56              signo2 = bivector(v23,v20)
57              signo3 = bivector(v31,v30)
58
59              //Se que los signos sean iguales, si son iguales implica que el
                    punto q se encuentra dentro del triángulo
60              Si signo1 = signo2 y signo2 = signo3 entonces
61                  Imprimir "Si hay intersección"
62              si no entonces
63                  Imprmir "No hay intersección"
64              Fin si
65
66          si no entonces
67
68              Imprimir "No hay intersección"
69          Fin si
70      Fin si
71  Fin si
72
73 Fin procedimiento
```

Este algoritmo requiere mayor cantidad de pasos, en primera instancia se verifica si uno o dos puntos del triángulo se encuentra en el interior de la esfera, de ser afirmativo, implica que hay intersección, no obstante, si los tres puntos se encuentran en el interior no hay intersección, en caso de que no se cumpla ninguna de estas condiciones entonces los tres puntos del triángulo están fuera de la esfera.

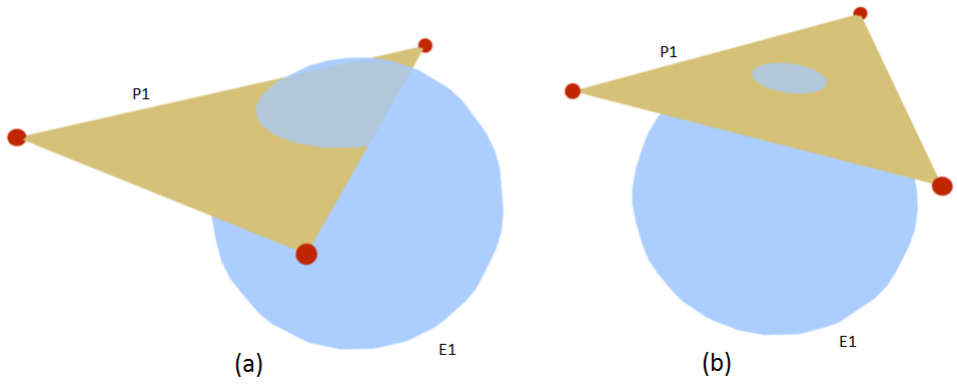


Figura IV.5: *Casos posibles en intersección Triángulo-Esfera*

Por lo tanto, se procede a calcular la intersección en el modelo conforme, si el plano intersecta a la esfera, entonces se realizan los siguientes pasos para determinar si el triángulo intersecta a la esfera.

El primer paso consiste en verificar si uno de los segmentos del triángulo intersecta a la esfera, en caso de ser afirmativo la colisión existe (figura IV.5, (a)), el caso más

complicado es cuando ningún segmento intersecta la esfera y aún así ocurre intersección (figura IV.5, (b)).

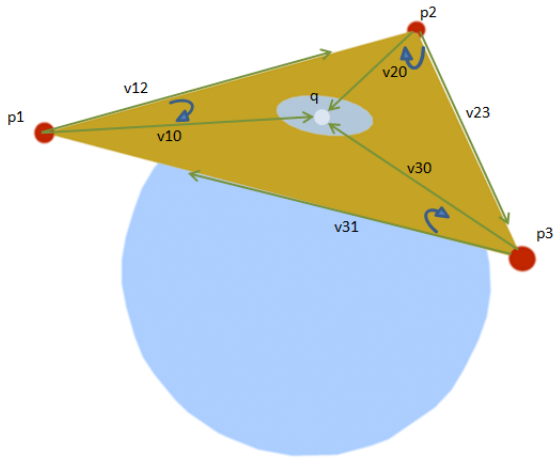


Figura IV.6: *Intersección Triangulo-Esfera*

Para solucionar el problema, se calcula el punto **q** (figura IV.6) cuya distancia sea mínima con respecto al centro, para verificar que este punto se encuentra en el interior del triángulo se aplica la técnica que se describe en el capítulo 1 sección 1.5.1 . Si el punto se encuentra en el interior entonces el triángulo intersecta la esfera, en caso contrario el triángulo no lo intersecta.

1.6 Intersección Esfera-Esfera

Algoritmo IV.6: Intersección esfera-esfera

```
1
2  Procedimiento InterseccionEsferaEsfera(E1, E2 )
3
4    //Intersección de la esfera E1 con la esfera E2
5
6     //Se coloca el origen en el centro de la esfera
7    CambioCoordenadasEsfera(E1)
8    CambioCoordenadasEsfera(E2)
9
10   r = CalculoInterseccionModeloConformeEsferaEsfera(E1,E2)
11
12    Si r >= 0 entonces
13      Imprimir "Si hay intersección"
14   si no entonces
15      Imprimir "No hay intersección"
16   Fin si
17
18 Fin procedimiento
```

Este algoritmo es más sencillo ya que solo es suficiente capturar el resultado de intersección en el modelo conforme.

2 Implementación

La implementación se realizó en dos fases.

- **Fase 1**: En esta fase se usó el programa Matlab para implementar los algoritmos, la elección de este programa se debe a la simplicidad para escribir algoritmos matemáticos.

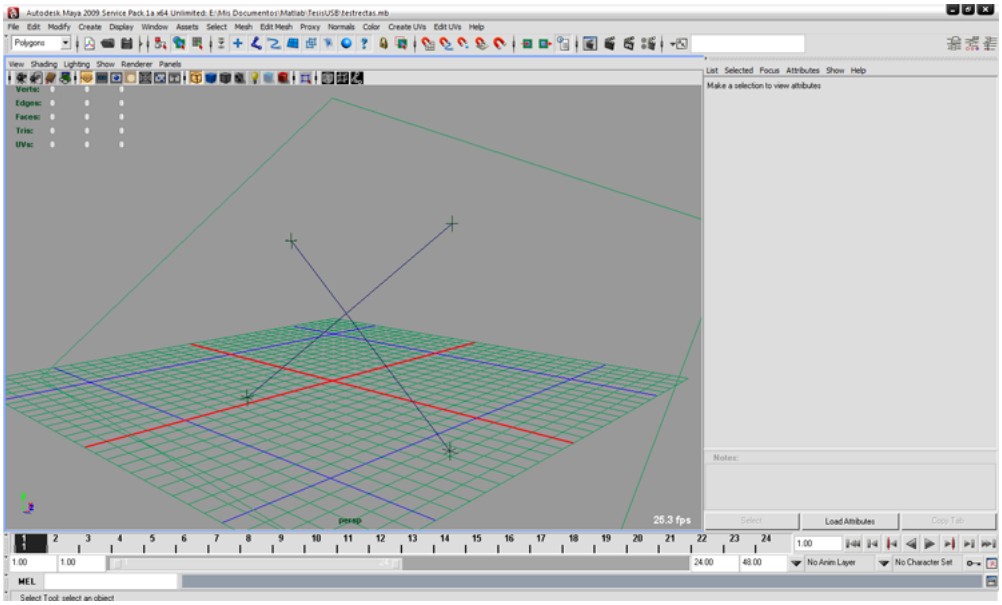


Figura IV.7: *Autodesk Maya*

También se uso el programa Autodesk Maya (programa de animación /modelado 3D), que fue de vital ayuda para probar los algoritmos para caso no triviales, por ejemplo, para probar si dos rectas se intersectaban, era complicado probar casos no triviales ya que había que realizar el cálculo manual, Maya evitaba este cálculo, ya que cuenta con manipuladores para moverse fácilmente en 3D.

Asimismo, se creo en Maya un código utilizando su lenguaje de programación MEL, que permite transferir la información desde Maya hasta Matlab. El uso de estos dos programas, hizo que la fase de prueba fuera precisa y rápida.

- **Fase 2**: Una vez que los algoritmos funcionaban en la fase 1, y se habían realizado las pruebas para casos no triviales, se procede a implementar los algoritmos en CPU y CUDA. En esta fase se utiliza lo siguiente:

  - Para implementar los algoritmos se utilizó Microsoft Visual C++ 2008.

- Para la interfaz se empleo los controladores de NVIDIA (NVIDIA Widgets) (http://code.google.com/p/nvidia-widgets/).
- Para cargar las mallas poligonales se uso la librería Trimesh [57].
- CPU: se utilizó un CPU 3.0Ghz Intel.
- GPU: se usó una NVIDIA 9600GT con 64 núcleos.

Esta fase fluyo con rapidez, ya que los algoritmos habían sido probados en la fase 1, y en el caso de un resultado no esperado o errado, era sencillo detectar el error, ya que se corrían los algoritmos en Matlab y se comparaban los resultados con los obtenidos en C++.

2.1 Programa

El programa usa la librería Opengl para el despliegue 3D, este consta de un menú (presionar el botón izquierdo del mouse) que presenta las opciones que tiene el programa.

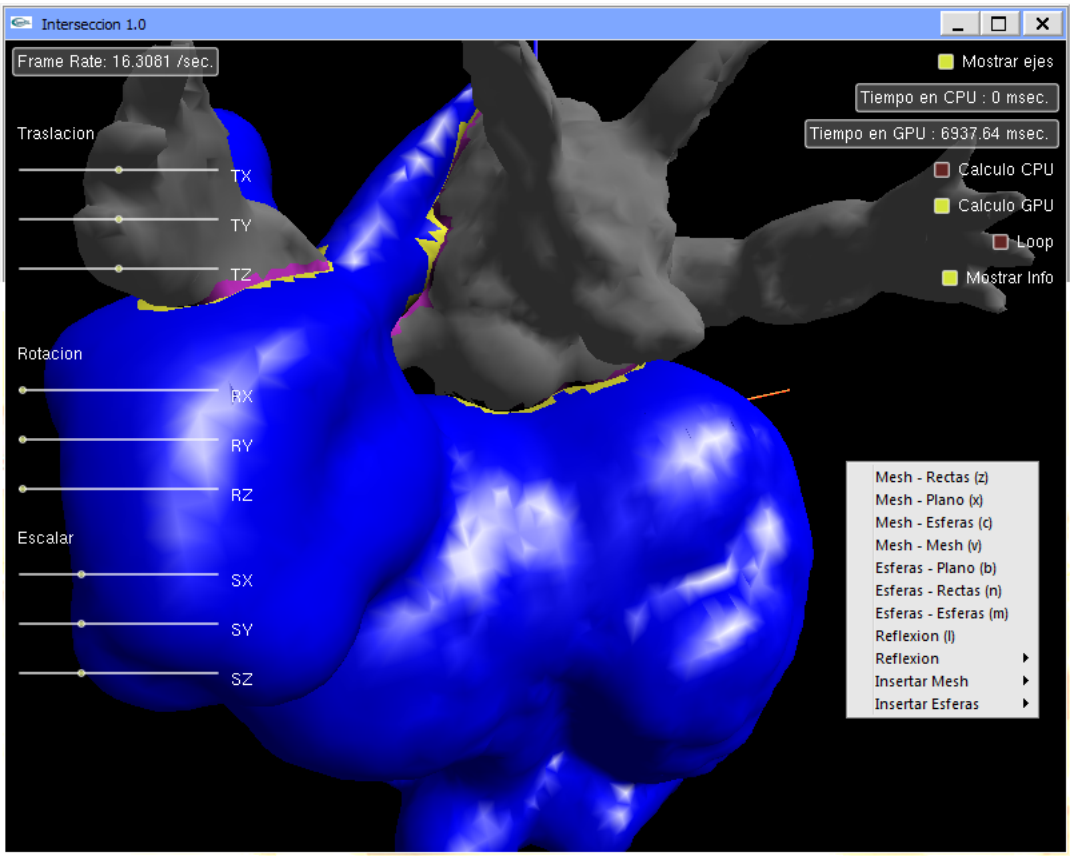


Figura IV.8: *Interfaz del Programa*

Las teclas para realizar las operaciones de intersección, se muestran en el menú entre los ( ) de cada opción. También se tienen las siguientes acciones:

- Tecla 1,2,...9: permite seleccionar la malla (mesh) que se allá introducido, los números corresponden al orden en que estas fueron insertadas.
- Tecla **f**: Permite aislar las caras que presentan alguna intersección.
- Tecla **d**: Elimina la malla seleccionada.

2.2 Configuración CUDA

Como se explica en el capítulo 3, es trabajo del programador proporcionar la cantidad de hilos y bloques que usa el código de CUDA para su ejecución, la interrogante sería ¿Como definir la cantidad de hilos para cada bloque?, dentro de las especificaciones de CUDA se encuentra que un bloque soporta un máximo de 512 hilos, no obstante, hay ciertas limitaciones.

Un multiprocesador o GPU core, puede almacenar 8192 registros, cuando se compila el programa este muestra que cada función de intersección necesita 32 registros. Por ejemplo, si se quiere intersectar una malla con una recta, la función de intersección necesita 32 registros (por hilo), por lo tanto, se si divide $8192/32 = 256$ hilos por bloque. En conclusión, para ejecutar cualquier tipo de intersección se usan 256 hilos por bloque.

En este programa un hilo equivale a un triángulo de la malla y un *kernel* es la función de intersección, como se tiene un máximo de 65536 bloques, se pueden procesar mallas de 16.000.000 millones de triángulos, cabe mencionar que no se esta trabajando con mallas de esta magnitud.

Para los cálculos de las transformaciones básicas (traslación, rotación y escalado) por ser operaciones más simples, la cantidad de registro utilizado es de 14 por hilo. Esto permite usar la máxima capacidad de los bloques, no obstante, se usaron 500 hilos por bloque, siguiendo la recomendación de la documentación de NVIDIA que señala de no llevar al límite el máximo de hilos permitidos.

2.3 Detalles de implementación

En el desarrollo del programa hubo algunos problemas que serán descritos brevemente a continuación:

- **Normalizar las rectas y planos**: en la fase 1 se detectó un problema de precisión, por ejemplo, supongamos que dos rectas se intersectan, si se toman los 4 puntos originales (2 por cada recta), el algoritmo no presentaba una buena precisión (en algunos casos) para igualar $eer = 0$, las pruebas revelaron que la razón era la distancia de los puntos, si se tomaba el punto de una de las rectas y se alejaba del otro punto una distancia considerable, la precisión disminuía.

Para hacer que el resultado numérico fuera preciso y no cambiara a medida que uno de los puntos se moviera, se optó por normalizar la recta.

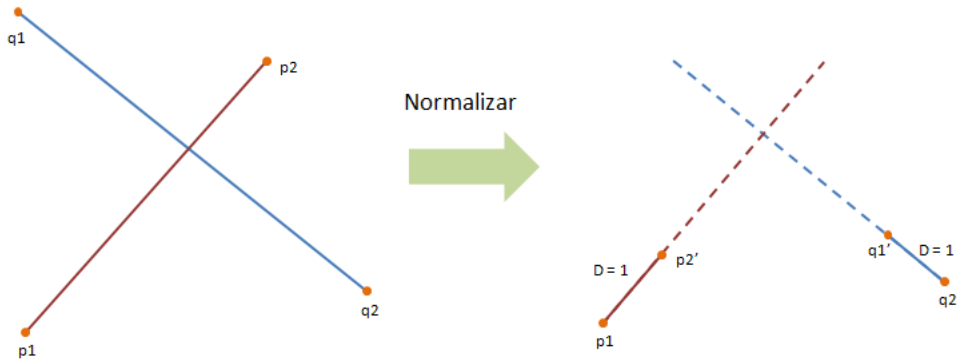


Figura IV.9: *Normalizar puntos de la recta*

Este proceso consiste en mover uno de los puntos de la recta tal que la longitud del segmento fuera igual a 1 (figura IV.9), con este arreglo, se tiene un valor numérico preciso y útil para validar los resultados.

- **Esferas**: Un problema similar sucede con las esferas, en el caso de intersección esfera-recta y esfera-plano, la solución fue centrar el origen en la esfera, pero en el caso de esfera-esfera, se tenia otro problema.

  Para intersectar dos esferas se toma el signo de B^2, tal como se expresa en el capítulo 2, sección 2.3.3, fórmula (2.1), en la fase 1 no se presentó problemas con esta fórmula, todos los resultados eran lo esperado, el problema se presenta en la fase 2. Sea el siguiente ejemplo, dos esferas con la siguiente configuración:

  - Esfera 1: centro = [0 0 0], radio = 8
  - Esfera 2: centro = [558,93 206,351 38,258], radio = 8

  Calculando la intersección se tiene que $B^2 = -8{,}7272e + 021$, como se puede apreciar el signo es negativo, pero la expresión numérica es extremadamente grande. Esto ocasiona problemas en la fase 2, en este caso el inconveniente no era de precisión sino de arquitectura del hardware. El programa fue creado en 32Bits, la razón de ello se debe a que la tarjeta gráfica utilizada la NVIDIA 9600GT es de 32 Bits. CUDA implementa los 64 bits a partir de la tarjeta de vídeo geforce GT220.

  Una variable del tipo **double** de 32 Bits, puede llegar hasta un valor máximo de $-2{,}147{,}483{,}648..{,}2{,}147{,}483{,}647$, como se aprecia, es muy inferior al número obtenido en matlab, esto daba inconsistencia en los resultado de la fase 2, cuando las esferas están muy alejadas unas de otras.

Por los momentos debido a las limitaciones del hardware esta falla no fue solventada, pero se aclara que el algoritmo funciona, ya que en la fase 1 no presenta ningún inconveniente.

2.4 Tiempo de ejecución

En primera instancia todos los algoritmos mencionados arriba toman un tiempo constante C para ejecutarse, ya que los algoritmos son instrucciones que se ejecutan sin ninguna iteración.

Como se menciona en el inicio del capítulo, se esta trabajando en el peor de los casos, por ejemplo, en el caso de intersectar una recta con una malla geométrica de n triángulos se procede a aplicar el algoritmo de intersección a cada triángulo de la malla. Por lo tanto los tiempos de ejecución son los siguientes:

- Interseccion Recta-Malla (n triángulos): el algoritmo es de orden $O(n)$.
- Interseccion Plano-Malla (n triángulos): el algoritmo es de orden $O(n)$.
- Interseccion Esfera-Malla (n triángulos): el algoritmo es de orden $O(n)$.
- Interseccion Malla (n triángulos)-Malla (m triángulo): el algoritmo es de orden $O(nm)$.
- Interseccion Recta- k cantidad de esferas: el algoritmo es de orden $O(k)$.
- Interseccion Plano- k cantidad de esferas: el algoritmo es de orden $O(k)$.
- Interseccion k cantidad de esferas: el algoritmo es de orden $O(k^2)$.

Estos tiempos son para la ejecución a través del CPU, en el caso del GPU, como se corre en paralelo es cuestión de dividir entre el numero de GPU cores o núcleos, es decir, si el algoritmo corre en orden $O(n)$, corriendo en paralelo sería de orden $O(n/c)$ donde c son los cores o núcleos que ejecutan los algoritmos.

CAPÍTULO V

RESULTADOS

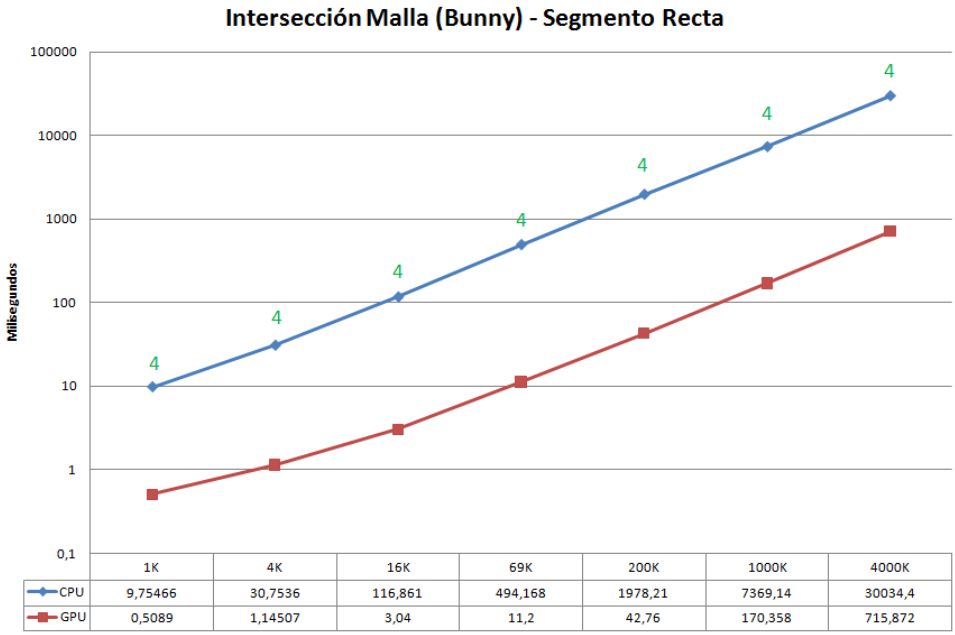
En un artículo de Daniel Fontijne y Leo Dorst [1], muestran el uso de distintas álgebras y geometrías en la computación gráfica, ellos aplican el estudio enfocándose en el Ray Tracing, donde efectivamente muestran que la Geometría Conforme 5D en tiempo de cómputo no es el más eficiente.

Consciente de esto, se quiere ver como el uso del GPU minimiza el tiempo de cómputo, en este caso como se vio en el capítulo 4, el estudio se enfoca al área de intersección o colisión de distintos elementos.

1 Intersección de primitivas con una malla poligonal

En las gráficas que se muestran a continuación se colisiona una malla poligonal con tres primitivas, segmento de recta, triángulo y esfera. El tiempo se mide en milisegundos y el valor en verde que se muestra en las gráficas representa los triángulos intersectados.

Para este estudio se uso la malla poligonal Bunny en diferentes resoluciones 1K, 4K, 16K, 69K, 200K, 1000K y 4000K, donde 1K = 1000 triángulos. En el apéndice B se muestran capturas de pantallas de las intersecciones.



	1K	4K	16K	69K	200K	1000K	4000K
CPU	9,75466	30,7536	116,861	494,168	1978,21	7369,14	30034,4
GPU	0,5089	1,14507	3,04	11,2	42,76	170,358	715,872

Figura V.1: *Intersección Segmento - Malla Bunny*

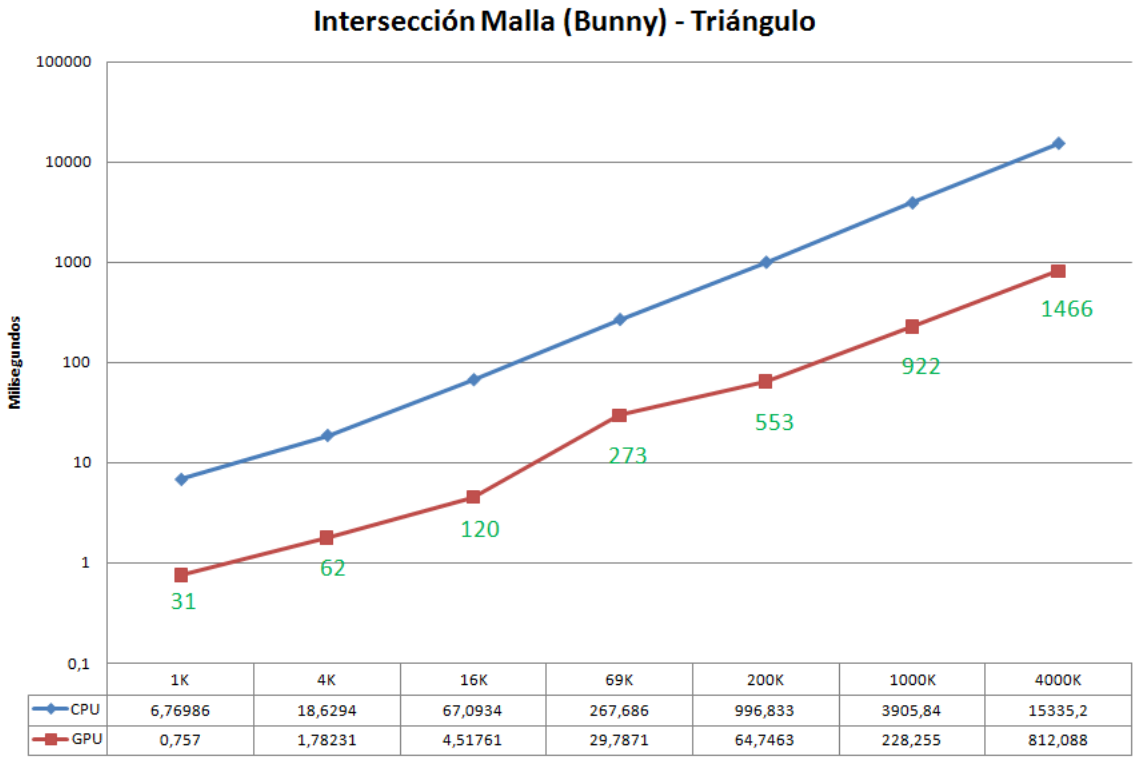


Figura V.2: *Intersección Triángulo - Malla Bunny*

1.1 Colisión de dos mallas poligonales

Aquí se muestran los tiempos de colisionar una malla poligonal 1K triángulos (Bunny), con diferentes resoluciones de otra malla poligonal (Armadillo).

Bunny - Armadillo	Bunny	Armadillo
1K - 5K	187	491
1K - 20K	185	996
1K - 100K	194	2215
1K - 345K	195	3968
1K - 1000K	107	5498
1K - 5000K	191	15063

La tabla muestra los triángulos intersectados por cada malla.

1.2 Colisión de Esferas con primitivas

En esta etapa se colisiona un conjunto de esferas (500, 1K, 2.5K, 5K y 10K) con las diferentes primitivas, para hacer esto, se generan esferas aleatorias en un volumen cuya medidas se expresa a continuación:

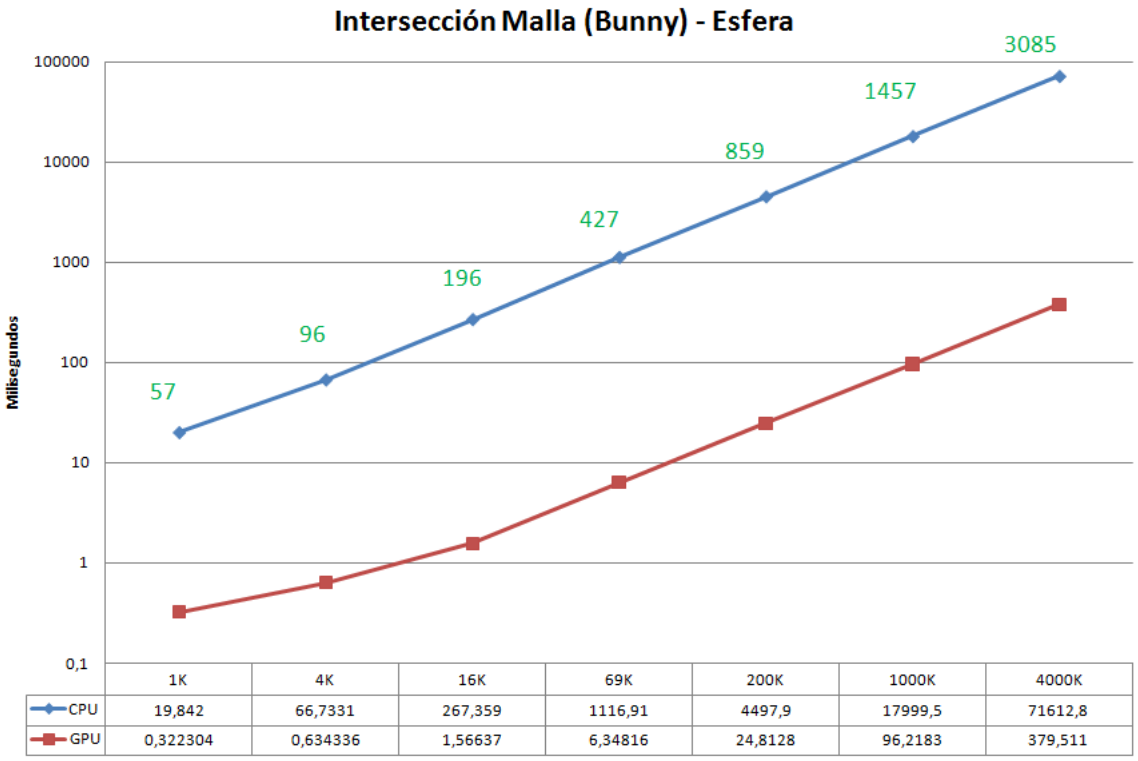


Figura V.3: *Intersección Esfera - Malla Bunny*

Esferas	Volumen
500	20x20x20
1K	20x20x20
2.5K	40x40x40
5K	80x80x80
10K	80x80x80

No obstante, por el problema descrito en el capítulo 4, que señala que si dos esferas están muy alejadas usando la plataforma de 32Bits, se podría tener un error en la validación de la intersección, solo para la intersección esferas con esferas los volúmenes para 5K y 10K esferas se cambio para 40x40x40, con este volumen el algoritmo funciona correctamente.

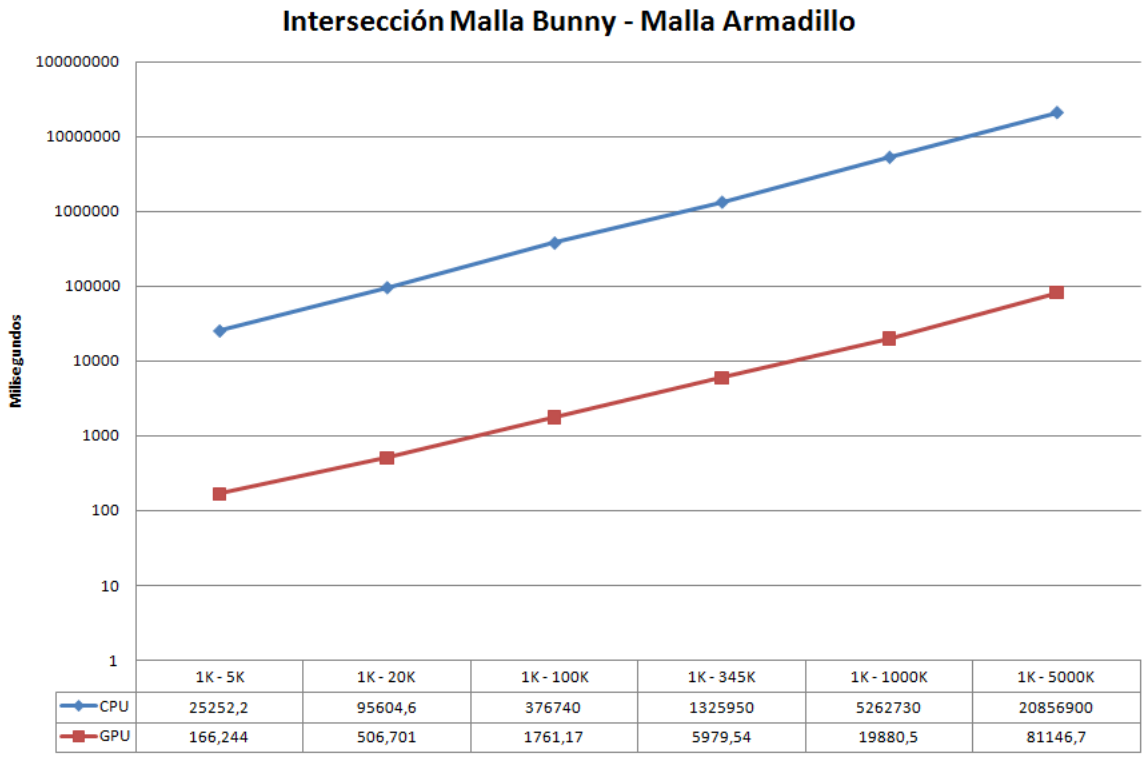


Figura V.4: *Intersección Malla Bunny - Malla Armadillo*

1.3 Análisis

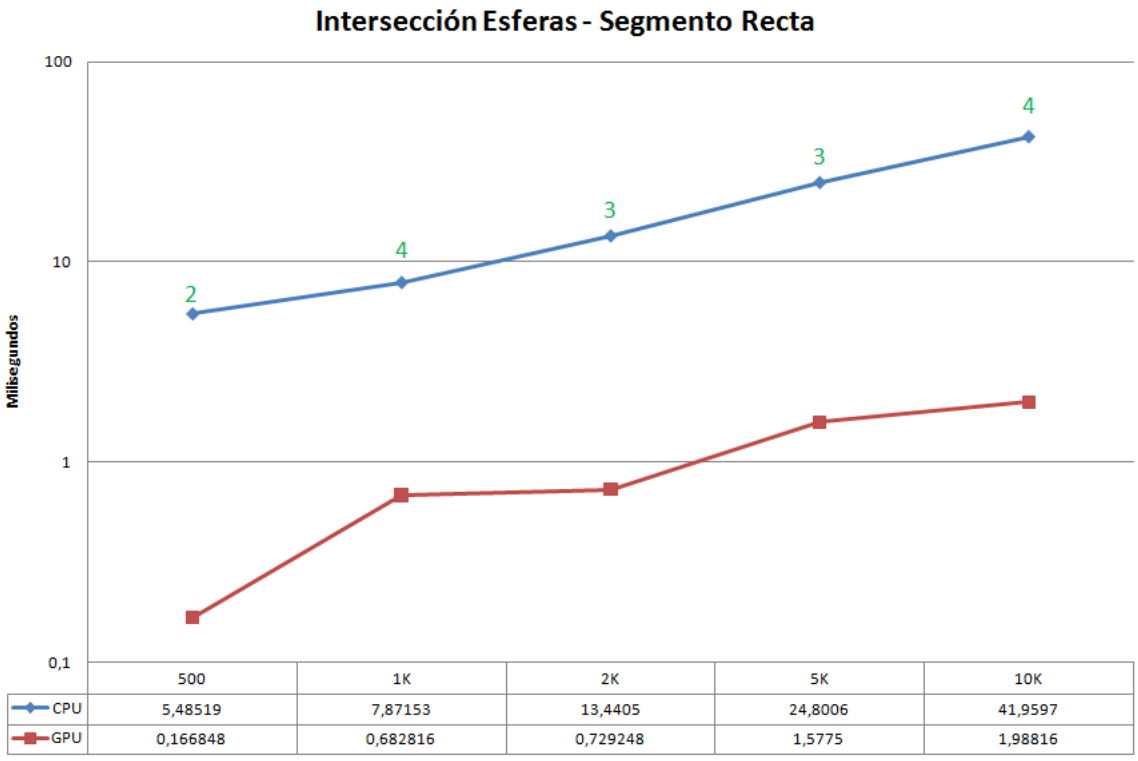
En primera instancia se deja en claro que las gráficas están en escala logarítmica. Las tres primeras gráficas (V.1,V.2 y V.3) muestran la intersección de una malla poligonal con las tres primitivas básicas: segmento de recta, triángulo y esferas. Como era de esperar el GPU reduce el tiempo de cómputo de manera muy radical, más aún, aunque cada colisión presenta tiempos diferentes se puede apreciar una misma tendencia en las gráficas.

Por ejemplo, tomando el último valor de la intersección Malla (Bunny) con el segmento de recta:

	Triángulos	Milisegundos	Segundos
CPU	4000K	30034,4	30,034
GPU	4000K	715,872	0,715

Se tiene que el GPU va 41X más rápido que el CPU, se puede apreciar que para 4000K el GPU ni siquiera alcanza el segundo de ejecución. Más aún, en las tres gráficas, cuando se intersectan las primitivas con la malla poligonal 4000K, ninguna supera el segundo de ejecución en el GPU, esto permite la posibilidad de hacer una ejecución del programa en tiempo real usando el GPU.

Los resultados de estas tres primeras gráficas, también muestran que la intersección malla (bunny) con un triángulo es mas rápido que la intersección de la misma malla con el



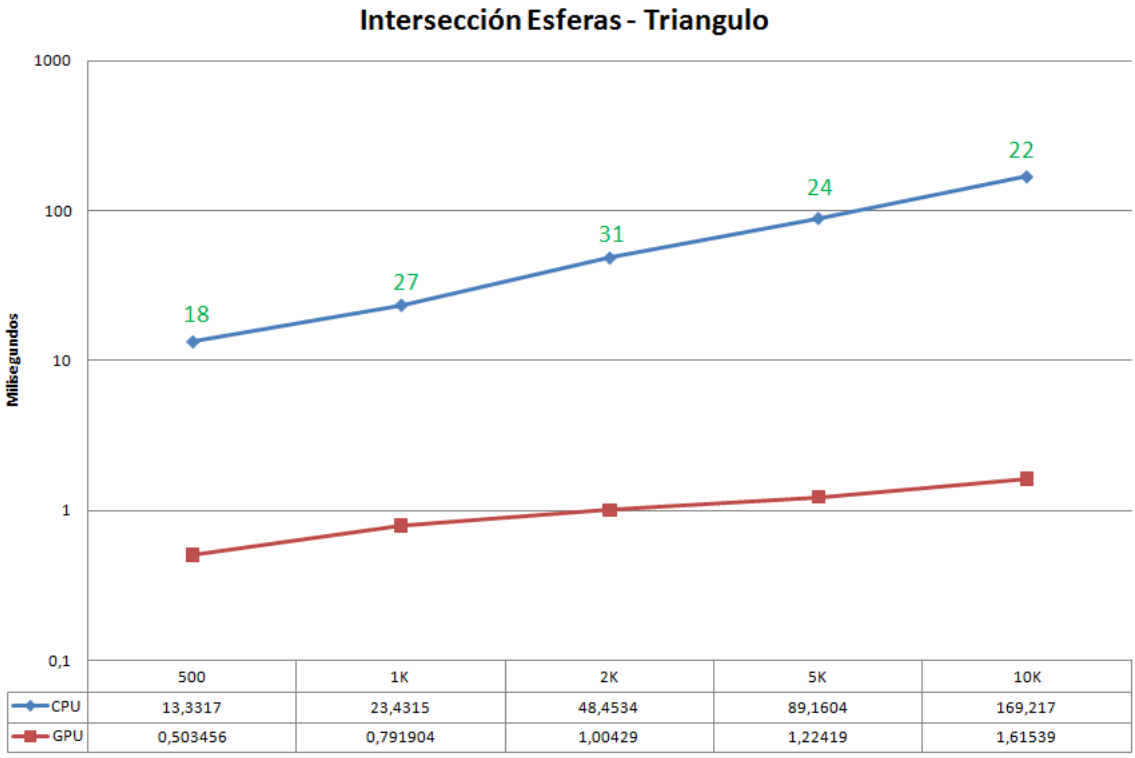
	500	1K	2K	5K	10K
CPU	5,48519	7,87153	13,4405	24,8006	41,9597
GPU	0,166848	0,682816	0,729248	1,5775	1,98816

Figura V.5: *Intersección Esferas - Segmento Recta*

segmento de recta y la esfera, esto en parte no se debe a que la fórmula de intersección sea más simple. Si se observa el Algoritmo 4.4 (línea 7), se tiene un pequeño filtro, es decir, si 3 puntos de un triángulo se encuentran de un lado del plano que forma el otro triángulo, el algoritmo termina su ejecución y no realiza los cálculos en el modelo conforme, ya que no existe la posibilidad de intersección, reduciendo el tiempo de ejecución considerablemente.

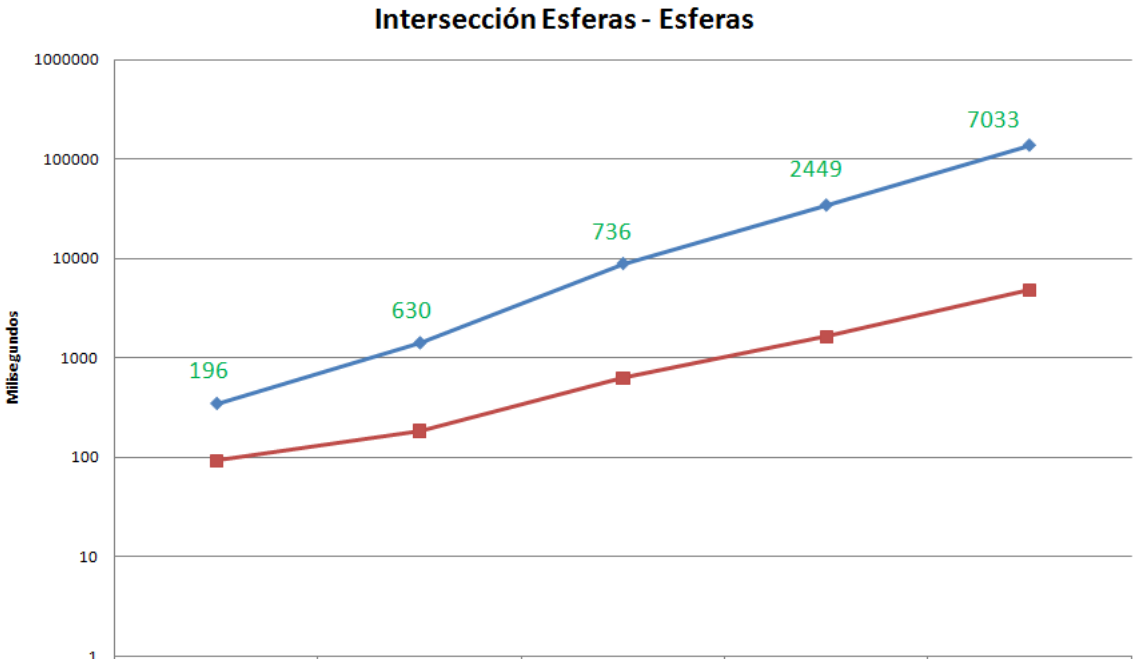
Luego se tiene la gráfica V.4, aquí se tiene un ejemplo muy claro del poder del GPU cuando se trabaja con una información de gran tamaño, se puede apreciar que la intersección de una malla poligonal de 4000K con otra malla de 1K, lo que genera 4.000.000.000 millones de iteraciones. Usando el CPU se genera un tiempo de cómputo bien alto (20856900 milisegundos = 20856,9 segundos = 347 minutos), si se compara con los 81 segundos que tarda en el GPU. Con esta gráfica realmente se puede ver la importancia actual del porque empresas diseñadoras de programas y científicos están usando el GPU como una alternativa muy atractiva para realizar cálculos numéricos.

Las últimas tres graficas V.5, V.6 y V.7 muestran la intersección de un grupo de esferas con las tres primitivas básicas, al igual que las gráficas anteriores se aprecia una diferencia considerable en los tiempos de cómputo entre CPU y el GPU, cuando se trabaja con una cantidad grandes de esferas (10K), inclusive en el caso de intersectar esferas con un segmento de recta o un triángulo, fácilmente el programa pudiera ejecutarse en tiempo real.



	500	1K	2K	5K	10K
CPU	13,3317	23,4315	48,4534	89,1604	169,217
GPU	0,503456	0,791904	1,00429	1,22419	1,61539

Figura V.6: *Intersección Esferas - Triángulo*



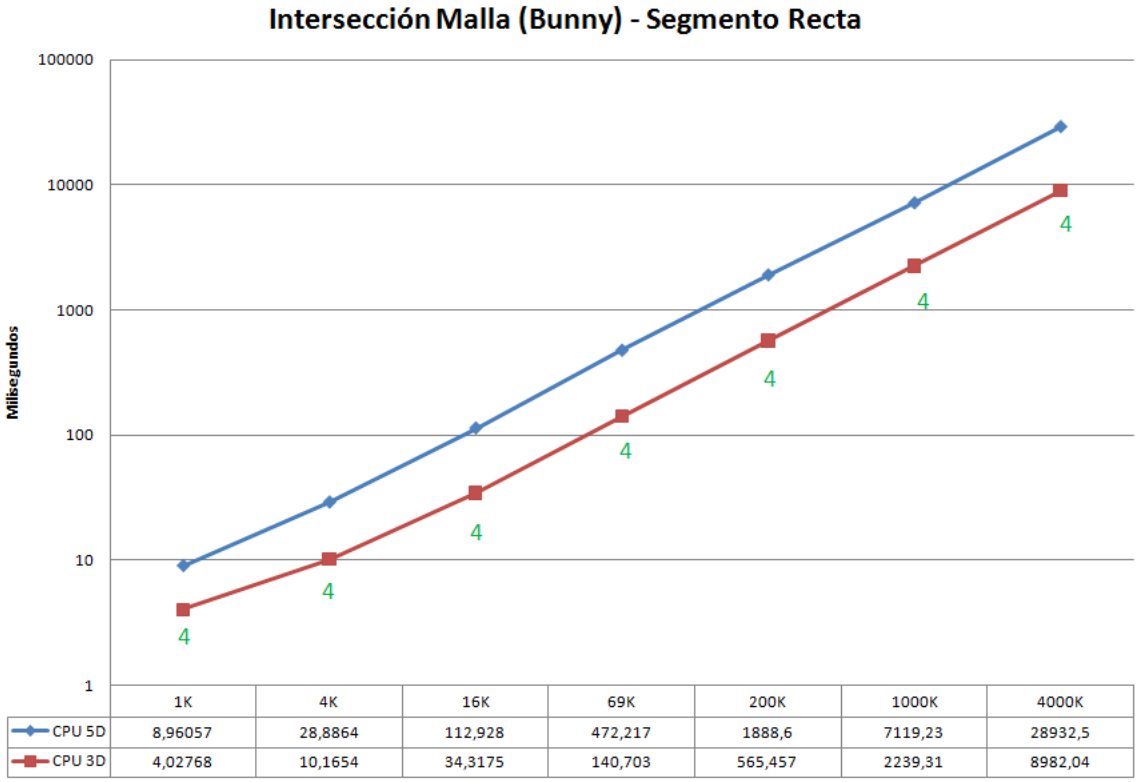
	500	1K	2K	5K	10K
CPU	343,649	1401,15	8789,35	34025,3	136377
GPU	92,8824	184,997	626,019	1632,56	4793,95

Figura V.7: *Intersección Esferas - Esferas*

2 Geometría Euclidea 3D vs Geometría Conforme 5D

Como un estudio extra, se compara a continuación la intersección de una malla con un segmento de recta y un triángulo, usando la Geometría Euclidea 3D, es claro, por el estudio hecho por Fontijne [1], que el modelo conforme es más lento, no obstante se quiere ver que tanto es la diferencia cuando se trabaja con intersecciones.

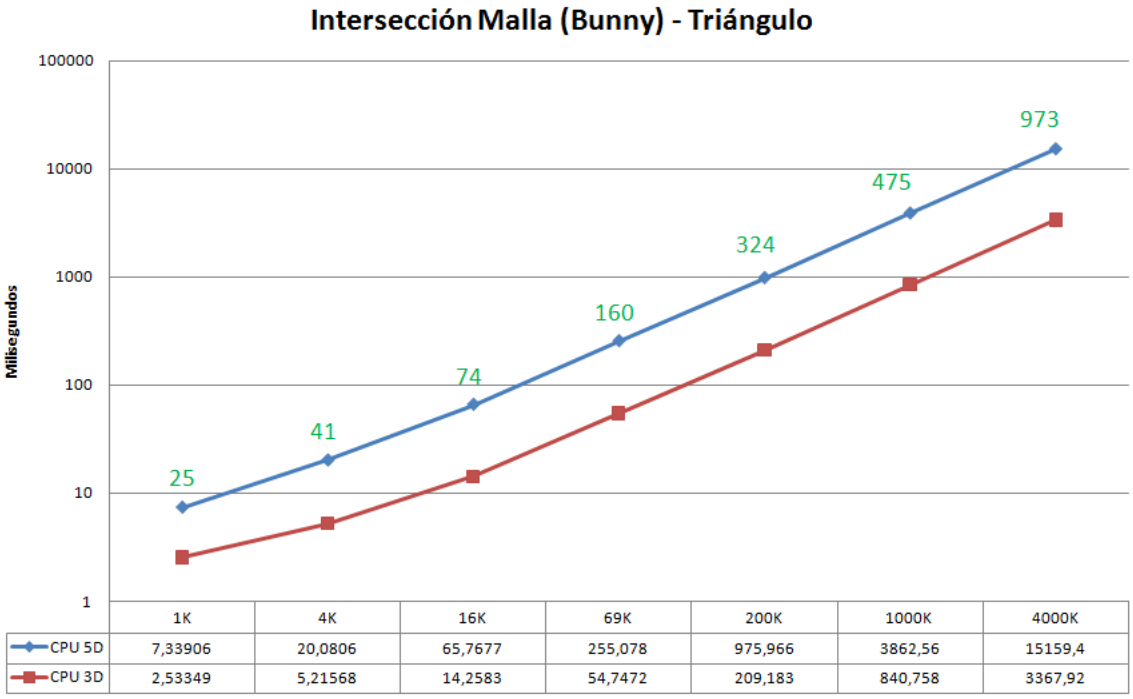
Para este estudio se uso solamente el CPU, y para los algoritmos en 3D se utilizaron los dos algoritmos más conocidos en términos de intersección, como son **A Fast Triangle-Triangle Intersection Test** de Tomas Moller [58] y **Fast, Minimum Storage Ray-Triangle Intersection** también de Tomas Moller y Ben Trumbore [59].



	1K	4K	16K	69K	200K	1000K	4000K
CPU 5D	8,96057	28,8864	112,928	472,217	1888,6	7119,23	28932,5
CPU 3D	4,02768	10,1654	34,3175	140,703	565,457	2239,31	8982,04

Figura V.8: *Intersección Malla Bunny - Segmento Recta*

Se aprecia en las gráficas V.8 y V.9, que el algoritmo en 3D es más eficiente que el algoritmo en el modelo conforme en 5D, se pudiera decir que es 2X o 4X más rápido que el modelo conforme. No obstante, hay que mencionar que el modelo conforme es algo que se esta empezando a usar en el mundo de la computación gráfica de manera reciente, con esto se quiere decir, que futuros trabajos pudieran optimizar los cálculos para que estos algoritmos sean más eficiente.



	1K	4K	16K	69K	200K	1000K	4000K
CPU 5D	7,33906	20,0806	65,7677	255,078	975,966	3862,56	15159,4
CPU 3D	2,53349	5,21568	14,2583	54,7472	209,183	840,758	3367,92

Figura V.9: *Intersección Malla Bunny - Triángulo*

CAPÍTULO VI

CONCLUSIÓN

Este trabajo presentó dos resultados importantes, el primero es referente a las fórmulas de intersección que se obtuvieron usando el modelo conforme, con estas fórmulas se pudo escribir un algoritmo para intersectar las primitivas básicas usadas en la computación gráficas. Se pudo apreciar como este modelo conforme simplifica la notación, como por ejemplo, el hecho que para intersectar las primitivas en este espacio se usa prácticamente una sola fórmula, como se describe en el capítulo 2. No obstante, por lo descrito en el apéndice A, se puede apreciar que realizar los cálculos en este espacio puede ser una tarea ardua.

El segundo punto importante, son los resultados obtenidos con estas intersecciones, donde se pudo comparar los tiempos de cómputo entre el CPU y el GPU. Con esos resultados se pudo ver como el GPU mejora considerablemente el tiempo de respuesta del algoritmo. Las pruebas indicaron que el GPU incrementaba la rapidez considerablemente hasta 41X en el caso de la intersección de una malla de triángulo con una recta.

Esto no significa que el uso de este espacio sea eficiente en término de cómputo. Al final del capítulo 5 se aprecia como este modelo es más lento que los algoritmos que usan el espacio euclídeo, no obstante, cabe mencionar que dicho espacio se esta aplicando de manera reciente en la computación gráfica. Este espacio por el hecho de estar en 5D, contiene mucha información para ser explorada. Como por ejemplo, cada intersección da como resultado un multivector, sería interesante indagar en trabajos futuros que representan esos valores en dicho multivector.

Cabe destacar que el tiempo de cómputo puede ser reducido considerablemente en el modelo conforme, si se emplean algunas estructura de datos que simplifique el proceso de intersección, como por ejemplo, Octree, BSP, entre otros, los algoritmos reducirían el tiempo de cómputo considerablemente, ya que no se estaría intersectando con cada triángulo de la malla poligonal.

También para trabajos futuros se puede considerar encontrar las fórmulas para encontrar los puntos, planos, rectas y círculos de cada intersección que no fueron tratados en este trabajo.

Como nota final este trabajo de grado presentó en un artículo [60], un resumen para la conferencia V Ibero-American Symposium in Computer Graphics, SIACG2011 a realizarse en Algarve, Portugal. Dicho artículo fue aprobado para ser presentado en la conferencia.

APÉNDICE A

INTERSECCIÓN Y TRANSFORMACIONES

En este apéndice se presentan los cálculos desarrollados para resolver los problemas de los objetivos planteados en la tesis.

1 Primitivas

En esta parte, se resuelve las ecuaciones de cada primitiva, hasta poner la ecuación en término de la base del espacio conforme. Para ello se considera la siguiente regla:

$$P_i = 2x_i + x_i^2 n - \bar{n} \tag{A.1}$$

Para desarrollar los cálculos se emplea hasta un máximo de 4 puntos x_i, los cuales se definen:

$$\begin{aligned} x_1 &= a_1 e_1 + a_2 e_2 + a_3 e_3 \\ x_2 &= b_1 e_1 + b_2 e_2 + b_3 e_3 \\ x_3 &= c_1 e_1 + c_2 e_2 + c_3 e_3 \\ x_4 &= d_1 e_1 + d_2 e_2 + d_3 e_3 \end{aligned} \tag{A.2}$$

1.1 Recta

Se tiene que una recta es igual:

$$L = P_1 \wedge P_2 \wedge n \tag{A.3}$$

Se calcula primero $P_1 \wedge P_2$,

$$\begin{aligned} P_1 \wedge P_2 &= (2x_1 + x_1^2 n - \bar{n}) \wedge (2x_2 + x_2^2 n - \bar{n}) \wedge n \\ P_1 \wedge P_2 &= 4(x_1 \wedge x_2) + 2x_2^2(x_1 \wedge n) - 2(x_1 \wedge \bar{n}) + 2x_1^2(n \wedge x_2) + x_1^2 x_2^2(n \wedge n) \\ &\quad - x_1^2(n \wedge \bar{n}) - 2(\bar{n} \wedge x_2) - x_2^2(\bar{n} \wedge n) + (\bar{n} \wedge n) \end{aligned} \tag{A.4}$$

Simplificando $n \wedge n = \bar{n} \wedge \bar{n} = 0$ se tiene:

$$\begin{aligned} P_1 \wedge P_2 &= 4(x_1 \wedge x_2) + 2x_2^2(x_1 \wedge n) - 2(x_1 \wedge \bar{n}) + 2x_1^2(n \wedge x_2) - x_1^2(n \wedge \bar{n}) \\ &\quad - 2(\bar{n} \wedge x_2) - x_2^2(\bar{n} \wedge n) \end{aligned} \tag{A.5}$$

Se calcula $L = P_1 \wedge P_2 \wedge n$,

$$\begin{aligned} L = {} & 4(x_1 \wedge x_2 \wedge n) + 2x_2^2(x_1 \wedge n \wedge n) - 2(x_1 \wedge \bar{n} \wedge n) + 2x_1^2(n \wedge x_2 \wedge n) \\ & - x_1^2(n \wedge \bar{n} \wedge n) - 2(\bar{n} \wedge x_2 \wedge n) - x_2^2(\bar{n} \wedge n \wedge n) \end{aligned} \tag{A.6}$$

Los factores donde se repite n se anulan,

$$\begin{aligned} L &= 4(x_1 \wedge x_2 \wedge n) - 2(x_1 \wedge \bar{n} \wedge n) + 2(x_2 \wedge \bar{n} \wedge n) \\ &= 4(x_1 \wedge x_2 \wedge n) + 2(x_1 \wedge n \wedge \bar{n}) - 2(x_2 \wedge n \wedge \bar{n}) \\ &= 4(x_1 \wedge x_2 \wedge n) - 2(x_2 - x_1) \wedge n \wedge \bar{n} \end{aligned} \tag{A.7}$$

Ahora se procede a expresar (A.7) en términos de la base del espacio conforme. Dicho cálculo se hace por segmentos empezando con $(x_1 \wedge x_2 \wedge n)$,

$$x_1 \wedge x_2 = (a_1b_2 - a_2b_1)e_1 \wedge e_2 + (a_2b_3 - a_3b_2)e_2 \wedge e_3 + (a_3b_1 - a_1b_3)e_3 \wedge e_1 \tag{A.8}$$

$$x_1 \wedge x_2 \wedge n = (a_1b_2 - a_2b_1)e_1 \wedge e_2 \wedge n + (a_2b_3 - a_3b_2)e_2 \wedge e_3 \wedge n + (a_3b_1 - a_1b_3)e_3 \wedge e_1 \wedge n$$

Ahora se expresa $(x_1 \wedge n \wedge \bar{n})$,

$$\begin{aligned} x_1 \wedge n \wedge \bar{n} &= (a_1e_1 + a_2e_2 + a_3e_3) \wedge n \wedge \bar{n} \\ &= a_1e_1 \wedge n \wedge \bar{n} + a_2e_1 \wedge n \wedge \bar{n} + a_3e_3 \wedge n \wedge \bar{n} \end{aligned} \tag{A.9}$$

Ahora se expresa $(x_2 \wedge n \wedge \bar{n})$,

$$\begin{aligned} x_2 \wedge n \wedge \bar{n} &= (b_1e_1 + b_2e_2 + b_3e_3) \wedge n \wedge \bar{n} \\ &= b_1e_1 \wedge n \wedge \bar{n} + b_2e_1 \wedge n \wedge \bar{n} + b_3e_3 \wedge n \wedge \bar{n} \end{aligned} \tag{A.10}$$

Tomando la resta de (A.7), $2(x_1 \wedge n \wedge \bar{n}) - 2(x_2 \wedge n \wedge \bar{n})$,

$$\begin{aligned} 2(x_1 \wedge n \wedge \bar{n}) - 2(x_2 \wedge n \wedge \bar{n}) = {} & 2[(a_1 - b_1)e_1 \wedge n \wedge \bar{n} + (a_2 - b_2)e_2 \wedge n \wedge \bar{n} \\ & + (a_3 - b_3)e_3 \wedge n \wedge \bar{n}] \end{aligned} \tag{A.11}$$

Por lo tanto L,

$$\begin{aligned} L = {} & 4[(a_1b_2 - a_2b_1)e_1 \wedge e_2 \wedge n + (a_2b_3 - a_3b_2)e_2 \wedge e_3 \wedge n + (a_3b_1 - a_1b_3)e_3 \wedge e_1 \wedge n] \\ & + 2[(a_1 - b_1)e_1 \wedge n \wedge \bar{n} + (a_2 - b_2)e_2 \wedge n \wedge \bar{n} + (a_3 - b_3)e_3 \wedge n \wedge \bar{n}] \end{aligned} \tag{A.12}$$

Ahora $n = e + \bar{e}$ y $n = e - \bar{e}$, por lo tanto (A.12),

$$\begin{aligned} L = 4[&(a_1b_2 - a_2b_1)e_1e_2e + (a_1b_2 - a_2b_1)e_1e_2\bar{e}+ \\ &(a_2b_3 - a_3b_2)e_2e_3e + (a_2b_3 - a_3b_2)e_2e_3\bar{e}+ \\ &(a_3b_1 - a_1b_3)e_3e_1e + (a_3b_1 - a_1b_3)e_3e_1\bar{e}]+ \\ &2[-2(a_1 - b_1)e_1e\bar{e} - 2(a_2 - b_2)e_2e\bar{e} - 2(a_3 - b_3)e_3e\bar{e}] \end{aligned} \tag{A.13}$$

Donde la ecuación final L queda igual,

$$\begin{aligned} L = 4[&(a_1b_2 - a_2b_1)e_1e_2e + (a_1b_2 - a_2b_1)e_1e_2\bar{e} \\ &+ (a_2b_3 - a_3b_2)e_2e_3e + (a_2b_3 - a_3b_2)e_2e_3\bar{e} \\ &+ (a_3b_1 - a_1b_3)e_3e_1e + (a_3b_1 - a_1b_3)e_3e_1\bar{e}] \\ &- (a_1 - b_1)e_1e\bar{e} - (a_2 - b_2)e_2e\bar{e} - (a_3 - b_3)e_3e\bar{e}] \end{aligned} \tag{A.14}$$

Por motivos de simplificar la fórmula para cálculos futuros, se define lo siguiente,

$$\begin{aligned} \beta_1 &= (a_1b_2 - a_2b_1) \\ \beta_2 &= (a_2b_3 - a_3b_2) \\ \beta_3 &= (a_3b_1 - a_1b_3) \\ \beta_4 &= (a_1 - b_1) \\ \beta_5 &= (a_2 - b_2) \\ \beta_6 &= (a_3 - b_3) \end{aligned} \tag{A.15}$$

Por lo tanto se simplifica la expresión de la recta L,

$$\begin{aligned} L = 4[&\beta_1e_1e_2e + \beta_1e_1e_2\bar{e} + \beta_2e_2e_3e + \beta_2e_2e_3\bar{e} + \beta_3e_3e_1e + \beta_3e_3e_1\bar{e} \\ &- \beta_4e_1e\bar{e} - \beta_5e_2e\bar{e} - \beta_6e_3e\bar{e}] \end{aligned} \tag{A.16}$$

1.2 Plano

Un plano se define:

$$L = P_1 \wedge P_2 \wedge P_3 \wedge n \tag{A.17}$$

De (A.5) tenemos:

$$\begin{aligned} P_1 \wedge P_2 = &4(x_1 \wedge x_2) + 2x_2^2(x_1 \wedge n) - 2(x_1 \wedge \bar{n}) + 2x_1^2(n \wedge x_2) - x_1^2(n \wedge \bar{n}) \\ &- 2(\bar{n} \wedge x_2) + x_2^2(n \wedge \bar{n}) \end{aligned} \tag{A.18}$$

Se procede a calcular $P_1 \wedge P_2 \wedge P_3$:

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 = (&4(x_1 \wedge x_2) + 2x_2^2(x_1 \wedge n) - 2(x_1 \wedge \bar{n}) + 2x_1^2(n \wedge x_2) - x_1^2(n \wedge \bar{n}) \\ &- 2(\bar{n} \wedge x_2) + x_2^2(n \wedge \bar{n})) \wedge (2x_3 + x_3^2 n - \bar{n}) \end{aligned} \tag{A.19}$$

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 =& 8(x_1 \wedge x_2 \wedge x_3) + 4x_3^2(x_1 \wedge x_2 \wedge n) - 4(x_1 \wedge x_2 \wedge \bar{n}) \\ &+ 4x_2^2(x_1 \wedge n \wedge x_3) + 2x_2^2 x_3^2(x_1 \wedge n \wedge n) - 2x_2^2(x_1 \wedge n \wedge \bar{n}) \\ &- 4(x_1 \wedge \bar{n} \wedge x_3) - 2x_3^2(x_1 \wedge \bar{n} \wedge n) + 2(x_1 \wedge \bar{n} \wedge \bar{n}) \\ &- 4x_1^2(x_2 \wedge n \wedge x_3) - 2x_1^2 x_3^2(x_2 \wedge n \wedge n) + 4x_1^2(x_2 \wedge n \wedge \bar{n}) \\ &- 2x_1(n \wedge \bar{n} \wedge x_3) - x_1^2 x_3^2(n \wedge \bar{n} \wedge n) + x_1^2(n \wedge \bar{n} \wedge \bar{n}) \\ &+ 4(x_2 \wedge \bar{n} \wedge x_3) + 2x_3^2(x_2 \wedge \bar{n} \wedge n) - 2(x_2 \wedge \bar{n} \wedge \bar{n}) \\ &+ 2x_2^2(n \wedge \bar{n} \wedge x_3) + x_2^2 x_3^2(n \wedge \bar{n} \wedge n) - x_2^2(n \wedge \bar{n} \wedge \bar{n}) \end{aligned} \tag{A.20}$$

Anulando los productos externos donde se repiten términos,

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 =& 8(x_1 \wedge x_2 \wedge x_3) + 4x_3^2(x_1 \wedge x_2 \wedge x_3) - 4(x_1 \wedge x_2 \wedge \bar{n}) \\ &+ 4x_2^2(x_1 \wedge n \wedge x_3) - 2x_2^2(x_1 \wedge n \wedge \bar{n}) \\ &- 4(x_1 \wedge \bar{n} \wedge x_3) - 2x_3^2(x_1 \wedge \bar{n} \wedge n) \\ &- 4x_1^2(x_2 \wedge n \wedge x_3) + 4x_1^2(x_2 \wedge n \wedge \bar{n}) \\ &- 2x_1(n \wedge \bar{n} \wedge x_3) + x_1^2(n \wedge \bar{n} \wedge \bar{n}) \\ &+ 4(x_2 \wedge \bar{n} \wedge x_3) + 2x_3^2(x_2 \wedge \bar{n} \wedge n) \\ &+ 2x_2^2(n \wedge \bar{n} \wedge x_3) \end{aligned} \tag{A.21}$$

Ahora se calcula $P_1 \wedge P_2 \wedge P_3 \wedge n$, además se retira las expresiones donde el producto externo contiene términos repetidos:

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 \wedge n =& 8(x_1 \wedge x_2 \wedge x_3 \wedge n) - 4(x_1 \wedge x_2 \wedge \bar{n} \wedge n) \\ &- 4(x_1 \wedge \bar{n} \wedge x_3 \wedge n) + 4(x_2 \wedge \bar{n} \wedge x_3 \wedge n) \end{aligned} \tag{A.22}$$

Ordenando los términos se tiene,

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 \wedge n =& 8(x_1 \wedge x_2 \wedge x_3 \wedge n) + 4[(x_1 \wedge x_2 \wedge n \wedge \bar{n}) \\ &- (x_1 \wedge x_3 \wedge n \wedge \bar{n}) + (x_2 \wedge x_3 \wedge n \wedge \bar{n})] \end{aligned} \tag{A.23}$$

Ahora se expresa esta última ecuación en términos de la base del espacio conforme, se resuelve primero la expresión $(x_1 \wedge x_2 \wedge x_3 \wedge n)$, usando (A.8) tenemos,

$$\begin{aligned} x_1 \wedge x_2 \wedge x_3 =& (a_1 b_2 c_3 - a_2 b_1 c_3) e_1 \wedge e_2 \wedge e_3 + (a_2 b_3 c_1 - a_3 b_2 c_1) e_1 \wedge e_2 \wedge e_3 \\ &+ (a_3 b_1 c_2 - a_1 b_3 c_2) e_1 \wedge e_2 \wedge e_3 \end{aligned} \tag{A.24}$$

Por lo tanto,

$$x_1 \wedge x_2 \wedge x_3 \wedge n = (a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_2b_1c_3 - a_3b_2c_1 - a_1b_3c_2)e_1 \wedge e_2 \wedge e_3 \wedge n$$

Resolviendo $(x_1 \wedge x_2 \wedge n \wedge \bar{n})$ se tiene,

$$\begin{aligned} x_1 \wedge x_2 \wedge n \wedge \bar{n} =& (a_1b_2 - a_2b_1)e_1 \wedge e_2 \wedge n \wedge \bar{n} + (a_2b_3 - a_3b_2)e_2 \wedge e_3 \wedge n \wedge \bar{n} \\ &+ (a_3b_1 - a_1b_3)e_3 \wedge e_1 \wedge n \wedge \bar{n} \end{aligned} \tag{A.25}$$

Resolviendo $(x_1 \wedge x_3 \wedge n \wedge \bar{n})$ se tiene,

$$\begin{aligned} x_1 \wedge x_3 \wedge n \wedge \bar{n} =& (a_1c_2 - a_2c_1)e_1 \wedge e_2 \wedge n \wedge \bar{n} + (a_2c_3 - a_3c_2)e_2 \wedge e_3 \wedge n \wedge \bar{n} \\ &+ (a_3c_1 - a_1c_3)e_3 \wedge e_1 \wedge n \wedge \bar{n} \end{aligned} \tag{A.26}$$

Y resolviendo $(x_2 \wedge x_3 \wedge n \wedge \bar{n})$ se tiene,

$$\begin{aligned} x_2 \wedge x_3 \wedge n \wedge \bar{n} =& (b_1c_2 - b_2c_1)e_1 \wedge e_2 \wedge n \wedge \bar{n} + (b_2c_3 - b_3c_2)e_2 \wedge e_3 \wedge n \wedge \bar{n} \\ &+ (b_3c_1 - b_1c_3)e_3 \wedge e_1 \wedge n \wedge \bar{n} \end{aligned} \tag{A.27}$$

Por lo tanto la ecuación (A.23) viene expresada por,

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 \wedge n =& 8[(a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_2b_1c_3 - a_3b_2c_1 - a_1b_3c_2)e_1 \wedge e_2 \wedge e_3 \wedge n] \\ &+ 4[(a_1b_2 - a_2b_1 - a_1c_2 + a_2c_1 + b_1c_2 - b_2c_1)e_1 \wedge e_2 \wedge n \wedge \bar{n} \\ &+ (a_2b_3 - a_3b_2 - a_2c_3 + a_3c_2 + b_2c_3 - b_3c_2)e_2 \wedge e_3 \wedge n \wedge \bar{n} \\ &+ (a_3b_1 - a_1b_3 - a_3c_1 + a_1c_3 + b_3c_1 - b_1c_3)e_3 \wedge e_1 \wedge n \wedge \bar{n}] \end{aligned} \tag{A.28}$$

Para simplificar los cálculos se toman las siguientes variables:

$$\begin{aligned} \omega_1 &= (a_1b_2c_3 + a_2b_3c_1 + a_3b_1c_2 - a_2b_1c_3 - a_3b_2c_1 - a_1b_3c_2) \\ \omega_2 &= (a_1b_2 - a_2b_1 - a_1c_2 + a_2c_1 + b_1c_2 - b_2c_1) \\ \omega_3 &= (a_2b_3 - a_3b_2 - a_2c_3 + a_3c_2 + b_2c_3 - b_3c_2) \\ \omega_4 &= (a_3b_1 - a_1b_3 - a_3c_1 + a_1c_3 + b_3c_1 - b_1c_3) \end{aligned} \tag{A.29}$$

Reemplazando n y $\bar{n}$ en término de las bases se obtiene,

$$P_1 \wedge P_2 \wedge P_3 \wedge n = 8\omega_1 e_1 \wedge e_2 \wedge e_3 \wedge n + 4[-2\omega_2 e_1e_2e\bar{e} - 2\omega_3 e_2e_3e\bar{e} - 2\omega_2 e_3e_1e\bar{e}] \tag{A.30}$$

Por último la ecuación del plano se expresa como,

$$P_1 \wedge P_2 \wedge P_3 \wedge n = 8[\omega_1 e_1e_2e_3e + \omega_1 e_1e_2e_3\bar{e} - \omega_2 e_1e_2e\bar{e} - \omega_3 e_2e_3e\bar{e} - \omega_2 e_3e_1e\bar{e}] \tag{A.31}$$

1.3 Esfera

Una esfera se define como:

$$E = P_1 \wedge P_2 \wedge P_3 \wedge P_4 \tag{A.32}$$

Ahora tomando el resultado (A.21), re-ordenando y aplicando factor común tenemos:

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 =& 8(x_1 \wedge x_2 \wedge x_3) + 4x_3^2(x_1 \wedge x_2 \wedge n) - 4(x_1 \wedge x_2 \wedge \bar{n}) \\ &+ 4x_1^2(x_2 \wedge x_3 \wedge n) - 4(x_2 \wedge x_3 \wedge \bar{n}) \\ &+ 4x_2^2(x_3 \wedge x_1 \wedge n) - 4(x_3 \wedge x_1 \wedge \bar{n}) \\ &+ (2x_3^2 - 2x_2^2)(x_1 \wedge n \wedge \bar{n}) \\ &+ (2x_1^2 - 2x_3^2)(x_2 \wedge n \wedge \bar{n}) \\ &+ (2x_2^2 - 2x_1^2)(x_3 \wedge n \wedge \bar{n}) \end{aligned} \tag{A.33}$$

Ahora se proceda a calcular la siguiente expresión,

$$\begin{aligned} P_1 \wedge P_2 \wedge P_3 \wedge P_4 =& [8(x_1 \wedge x_2 \wedge x_3) + 4x_3^2(x_1 \wedge x_2 \wedge n) - 4(x_1 \wedge x_2 \wedge \bar{n}) \\ &+ 4x_1^2(x_2 \wedge x_3 \wedge n) - 4(x_2 \wedge x_3 \wedge \bar{n}) \\ &+ 4x_2^2(x_3 \wedge x_1 \wedge n) - 4(x_3 \wedge x_1 \wedge \bar{n}) \\ &+ (2x_3^2 - 2x_2^2)(x_1 \wedge n \wedge \bar{n}) \\ &+ (2x_1^2 - 2x_3^2)(x_2 \wedge n \wedge \bar{n}) \\ &+ (2x_2^2 - 2x_1^2)(x_3 \wedge n \wedge \bar{n})] \wedge (2x_4 + x_4^2 n - \bar{n}) \end{aligned} \tag{A.34}$$

Por lo tanto,

$$\begin{aligned} E =& 16(x_1 \wedge x_2 \wedge x_3 \wedge x_4) + 8x_3^2(x_1 \wedge x_2 \wedge n \wedge x_4) - 8(x_1 \wedge x_2 \wedge \bar{n} \wedge x_4) \\ &+ 8x_1^2(x_2 \wedge x_3 \wedge n \wedge x_4) - 8(x_2 \wedge x_3 \wedge \bar{n} \wedge x_4) + 8x_2^2(x_3 \wedge x_1 \wedge n \wedge x_4) \\ &- 8(x_3 \wedge x_1 \wedge \bar{n} \wedge x_4) + (4x_3^2 - 4x_2^2)(x_1 \wedge n \wedge \bar{n} \wedge x_4) + (4x_1^2 - 4x_3^2)(x_1 \wedge n \wedge \bar{n} \wedge x_4) \\ &+ (4x_2^2 - 4x_1^2)(x_3 \wedge n \wedge \bar{n} \wedge x_4) + 8x_4^2(x_1 \wedge x_2 \wedge x_3 \wedge n) - 4x_4^2(x_1 \wedge x_2 \wedge \bar{n} \wedge n) \\ &- 4x_4^2(x_2 \wedge x_3 \wedge \bar{n} \wedge n) - 4x_4^2(x_3 \wedge x_1 \wedge \bar{n} \wedge n) - 8(x_1 \wedge x_2 \wedge x_3 \wedge \bar{n}) \\ &- 4x_3^2(x_1 \wedge x_2 \wedge n \wedge \bar{n}) - 4x_1^2(x_2 \wedge x_3 \wedge n \wedge \bar{n}) - 4x_2^2(x_3 \wedge x_1 \wedge n \wedge \bar{n}) \end{aligned} \tag{A.35}$$

Re-ordenando y haciendo $n = e + \bar{e}$ y $\bar{n} = e - \bar{e}$,

$$
\begin{aligned}
E =& 16(x_1 \wedge x_2 \wedge x_3 \wedge x_4) - 8x_3^2(x_1 \wedge x_2 \wedge x_4 \wedge e) - 8x_3^2(x_1 \wedge x_2 \wedge x_4 \wedge \bar{e}) \\
&+ 8(x_1 \wedge x_2 \wedge x_4 \wedge e) - 8(x_1 \wedge x_2 \wedge x_4 \wedge \bar{e}) - 8x_1^2(x_2 \wedge x_3 \wedge x_4 \wedge e) \\
&- 8x_1^2(x_2 \wedge x_3 \wedge x_4 \wedge \bar{e}) + 8(x_2 \wedge x_3 \wedge x_4 \wedge e) - 8(x_2 \wedge x_3 \wedge x_4 \wedge \bar{e}) \\
&- 8x_2^2(x_3 \wedge x_1 \wedge x_4 \wedge e) - -8x_2^2(x_3 \wedge x_1 \wedge x_4 \wedge \bar{e}) + 8(x_3 \wedge x_1 \wedge x_4 \wedge e) \\
&- 8(x_3 \wedge x_1 \wedge x_4 \bar{e}) + 8x_4^2(x_1 \wedge x_2 \wedge x_3 \wedge e) + 8x_4^2(x_1 \wedge x_2 \wedge x_3 \wedge \bar{e}) \\
&- 8(x_1 \wedge x_2 \wedge x_3 \wedge e) + 8(x_1 \wedge x_2 \wedge x_3 \wedge \bar{e}) - 8(x_3^2 - x_2^2)(x_1 \wedge x_4 \wedge e \wedge \bar{e}) \\
&- 8(x_1^2 - x_3^2)(x_2 \wedge x_4 \wedge e \wedge \bar{e}) - 8(x_2^2 - x_1^2)(x_3 \wedge x_4 \wedge e \wedge \bar{e}) \\
&- 8(x_4^2 - x_3^2)(x_1 \wedge x_2 \wedge e \wedge \bar{e}) - 8(x_4^2 - x_1^2)(x_2 \wedge x_3 \wedge e \wedge \bar{e}) \\
&- 8(x_4^2 - x_2^2)(x_3 \wedge x_1 \wedge e \wedge \bar{e})
\end{aligned} \tag{A.36}
$$

Usando el resultado (A.24) y (A.8) se tiene,

$$
\begin{aligned}
E =& \alpha_{124}(1 - x_3^2)(e_1 \wedge e_2 \wedge e_3 \wedge e) - \alpha_{124}(1 - x_3^2)(e_1 \wedge e_2 \wedge e_3 \wedge \bar{e}) \\
&+ \alpha_{234}(1 - x_1^2)(e_1 \wedge e_2 \wedge e_3 \wedge e) - \alpha_{234}(1 - x_1^2)(e_1 \wedge e_2 \wedge e_3 \wedge \bar{e}) \\
&+ \alpha_{314}(1 - x_2^2)(e_1 \wedge e_2 \wedge e_3 \wedge e) - \alpha_{314}(1 - x_2^2)(e_1 \wedge e_2 \wedge e_3 \wedge \bar{e}) \\
&+ \alpha_{123}(x_4^2 - 1)(e_1 \wedge e_2 \wedge e_3 \wedge e) - \alpha_{123}(x_4^2 - 1)(e_1 \wedge e_2 \wedge e_3 \wedge \bar{e}) \\
&- (x_3^2 - x_2^2)[ad_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + ad_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + ad_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})] \\
&- (x_1^2 - x_3^2)[bd_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + bd_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + bd_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})] \\
&- (x_2^2 - x_1^2)[cd_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + cd_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + cd_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})] \\
&- (x_4^2 - x_3^2)[ab_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + ab_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + ab_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})] \\
&- (x_4^2 - x_1^2)[bc_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + bc_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + bc_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})] \\
&- (x_4^2 - x_2^2)[ca_{12}(e_1 \wedge e_2 \wedge e \wedge \bar{e}) + ca_{23}(e_2 \wedge e_3 \wedge e \wedge \bar{e}) + ca_{31}(e_3 \wedge e_1 \wedge e \wedge \bar{e})]
\end{aligned}
$$

Donde,

$$
\begin{aligned}
\alpha_{124} &= (a_1 b_2 d_3 + a_2 b_3 d_1 + a_3 b_1 d_2 - a_2 b_1 d_3 - a_3 b_2 d_1 - a_1 b_3 d_2) \\
\alpha_{234} &= (b_1 c_2 d_3 + b_2 c_3 d_1 + b_3 c_1 d_2 - b_2 c_1 d_3 - b_3 c_2 d_1 - b_1 c_3 d_2) \\
\alpha_{314} &= (c_1 a_2 d_3 + c_2 a_3 d_1 + c_3 a_1 d_2 - c_2 a_1 d_3 - c_3 a_2 d_1 - c_1 a_3 d_2) \\
\alpha_{123} &= (a_1 b_2 c_3 + a_2 b_3 c_1 + a_3 b_1 c_2 - a_2 b_1 c_3 - a_3 b_2 c_1 - a_1 b_3 c_2)
\end{aligned} \tag{A.37}
$$

Y las variables del tipo ad_{12}, bc_{31}, etc, se expresan con la siguiente notación,

$$
xy_{ij} = x_i y_j - x_j y_i \tag{A.38}
$$

Factorizando y simplificando se tiene,

$$
E = \mu_1 e_1 e_2 e_3 e + \mu_2 e_1 e_2 e_3 \bar{e} - \mu_3 e_1 e_2 e \bar{e} - \mu_4 e_2 e_3 e \bar{e} - \mu_5 e_3 e_1 e \bar{e} \tag{A.39}
$$

Donde,

$$\begin{aligned}
\mu_1 =&\alpha_{124}(1-x_3^2)+\alpha_{234}(1-x_1^2)+\alpha_{314}(1-x_2^2)+\alpha_{123}(x_4^2-1)\\
\mu_2 =&\alpha_{123}(1+x_4^2)-\alpha_{124}(1+x_3^2)-\alpha_{234}(1+x_1^2)-\alpha_{314}(1+x_2^2)\\
\mu_3 =&ad_{12}(x_3^2-x_2^2)+bd_{12}(x_1^2-x_3^2)+cd_{12}(x_2^2-x_1^2)+ab_{12}(x_4^2-x_3^2)\\
&+bc_{12}(x_4^2-x_1^2)+ca_{12}(x_4^2-x_2^2)\\
\mu_4 =&ad_{23}(x_3^2-x_2^2)+bd_{23}(x_1^2-x_3^2)+cd_{23}(x_2^2-x_1^2)+ab_{23}(x_4^2-x_3^2)\\
&+bc_{23}(x_4^2-x_1^2)+ca_{23}(x_4^2-x_2^2)\\
\mu_5 =&ad_{31}(x_3^2-x_2^2)+bd_{31}(x_1^2-x_3^2)+cd_{31}(x_2^2-x_1^2)+ab_{31}(x_4^2-x_3^2)\\
&+bc_{31}(x_4^2-x_1^2)+ca_{31}(x_4^2-x_2^2)
\end{aligned} \tag{A.40}$$

2 Intersecciones

En los resultados obtenidos en la parte superior, al calcular las primitivas, se tiene un escalar que multiplica a una expresión, para efecto de calcular la intersección, dicho escalar no influye en el resultado que se quiere obtener, por lo cual se omite en las fórmulas de las intersecciones.

2.1 Recta-Recta

A continuación se procede a calcular la siguiente expresión:

$$B = (L_1 \vee L_2)^* = (IL_1)\cdot L_2 \tag{A.41}$$

De la ecuación (A.16) se tiene,

$$\begin{aligned}
L_1 =&\alpha_1 e_1e_2e+\alpha_1e_1e_2\bar{e}+\alpha_2e_2e_3e+\alpha_2e_2e_3\bar{e}+\alpha_3e_3e_1e+\alpha_3e_3e_1\bar{e}\\
&-\alpha_4e_1e\bar{e}-\alpha_5e_2e\bar{e}-\alpha_6e_3e\bar{e}
\end{aligned} \tag{A.42}$$

$$\begin{aligned}
L_2 =&\beta_1 e_1e_2e+\beta_1e_1e_2\bar{e}+\beta_2e_2e_3e+\beta_2e_2e_3\bar{e}+\beta_3e_3e_1e+\beta_3e_3e_1\bar{e}\\
&-\beta_4e_1e\bar{e}-\beta_5e_2e\bar{e}-\beta_6e_3e\bar{e}
\end{aligned} \tag{A.43}$$

Se calcula primero IL_1,

$$\begin{aligned}
IL_1 =&\alpha_1e_1e_2e_3e\bar{e}e_1e_2e+\alpha_1e_1e_2e_3e\bar{e}e_1e_2\bar{e}\\
&+\alpha_2e_1e_2e_3e\bar{e}e_2e_3e+\alpha_2e_1e_2e_3e\bar{e}e_2e_3\bar{e}\\
&+\alpha_3e_1e_2e_3e\bar{e}e_3e_1e+\alpha_3e_1e_2e_3e\bar{e}e_3e_1\bar{e}\\
&-\alpha_4e_1e_2e_3e\bar{e}e_1e\bar{e}-\alpha_5e_1e_2e_3e\bar{e}e_2e\bar{e}-\alpha_6e_1e_2e_3e\bar{e}e_3e\bar{e}
\end{aligned} \tag{A.44}$$

Para calcular este producto se usa lo siguiente:

$$\begin{aligned} e_1e_1 = e_2e_2 = e_3e_3 = ee = 1 \\ \bar{e}\bar{e} = -1 \\ e_ie_j = -e_je_i \end{aligned} \tag{A.45}$$

Por lo tanto,

$$\begin{aligned} IL_1 =& \alpha_1e_3\bar{e} + \alpha_1e_3e + \alpha_2e_1\bar{e} + \alpha_2e_1e + \alpha_3e_2\bar{e} + \alpha_3e_2e \\ & - \alpha_4e_2e_3 - \alpha_5e_3e_1 - \alpha_6e_1e_2 \end{aligned} \tag{A.46}$$

Ahora se calcula el producto interno $B = IL_1 \cdot L_2$,

$$\begin{aligned} B =& - \alpha_1\beta_1e_1e_2e_3 + \alpha_1\beta_2e_2e\bar{e} - \alpha_1\beta_2e_2 - \alpha_1\beta_3e_1e\bar{e} - \alpha_1\beta_3e_1 + \alpha_1\beta_4e_3e_1e \\ & - \alpha_1\beta_5e_2e_3e + \alpha_1\beta_6e + \alpha_1\beta_1e_1e_2e_3 - \alpha_1\beta_2e_2 - \alpha_1\beta_2e_2e\bar{e} + \alpha_1\beta_3e_1 \\ & + \alpha_1\beta_3e_1e\bar{e} + \alpha_1\beta_4e_3e_1\bar{e} - \alpha_1\beta_5e_2e_3\bar{e} + \alpha_1\beta_6\bar{e} - \alpha_2\beta_1e_2e\bar{e} - \alpha_2\beta_1e_2 \\ & - \alpha_2\beta_2e_1e_2e_3 + \alpha_2\beta_3e_3e\bar{e} + \alpha_2\beta_3e_3 + \alpha_2\beta_4e + \alpha_2\beta_5e_1e_2e - \alpha_2\beta_6e_3e_1e \\ & + \alpha_2\beta_1e_2 + \alpha_2\beta_1e_2e\bar{e} + \alpha_2\beta_2e_1e_2e_3 - \alpha_2\beta_3e_3 - \alpha_2\beta_3e_3e\bar{e} + \alpha_2\beta_4\bar{e} \\ & + \alpha_2\beta_5e_1e_2\bar{e} - \alpha_2\beta_6e_3e_1\bar{e} + \alpha_3\beta_1e_1e\bar{e} + \alpha_3\beta_1e_1 - \alpha_3\beta_2e_3e\bar{e} - \alpha_3\beta_2e_3 \\ & - \alpha_3\beta_3e_1e_2e_3 - \alpha_3\beta_4e_1e_2e + \alpha_3\beta_5e + \alpha_3\beta_6e_2e_3e - \alpha_3\beta_1e_1 - \alpha_3\beta_1e_1e\bar{e} \\ & + \alpha_3\beta_2e_3 + \alpha_3\beta_2e_3e\bar{e} + \alpha_3\beta_3e_1e_2e_3 - \alpha_3\beta_4e_1e_2\bar{e} + \alpha_3\beta_5\bar{e} + \alpha_3\beta_6e_2e_3\bar{e} \\ & - \alpha_4\beta_1e_3e_1e - \alpha_4\beta_1e_3e_1\bar{e} + \alpha_4\beta_2e + \alpha_4\beta_2\bar{e} + \alpha_4\beta_3e_1e_2e + \alpha_4\beta_3e_1e_2\bar{e} \\ & - \alpha_4\beta_5e_3e\bar{e} + \alpha_4\beta_6e_2e\bar{e} + \alpha_5\beta_1e_2e_3e + \alpha_5\beta_1e_2e_3\bar{e} - \alpha_5\beta_2e_1e_2e - \alpha_5\beta_2e_1e_2\bar{e} \\ & + \alpha_5\beta_3e + \alpha_5\beta_3\bar{e} + \alpha_5\beta_4e_3e\bar{e} - \alpha_5\beta_6e_1e\bar{e} + \alpha_6\beta_1e + \alpha_6\beta_1\bar{e} \\ & + \alpha_6\beta_2e_3e_1e + \alpha_6\beta_2e_3e_1\bar{e} - \alpha_6\beta_3e_2e_3e - \alpha_6\beta_3e_2e_3\bar{e} - \alpha_6\beta_4e_2e\bar{e} + \alpha_6\beta_5e_1e\bar{e} \\ & + \alpha_6\beta_3e_1e_2e_3 \end{aligned} \tag{A.47}$$

Simplificando se obtiene,

$$\begin{aligned} B =& (\alpha_2\beta_5 - \alpha_3\beta_4 + \alpha_4\beta_3 - \alpha_5\beta_2)e_1e_2e + (\alpha_3\beta_6 - \alpha_1\beta_5 + \alpha_5\beta_1 - \alpha_6\beta_3)e_2e_3e \\ & + (\alpha_1\beta_4 - \alpha_2\beta_6 + \alpha_6\beta_2 - \alpha_4\beta_1)e_3e_1e + (\alpha_2\beta_5 - \alpha_3\beta_4 + \alpha_4\beta_3 - \alpha_5\beta_2)e_1e_2\bar{e} \\ & + (\alpha_3\beta_6 - \alpha_1\beta_5 + \alpha_5\beta_1 - \alpha_6\beta_3)e_2e_3\bar{e} + (\alpha_1\beta_4 - \alpha_2\beta_6 + \alpha_6\beta_2 - \alpha_4\beta_1)e_3e_1\bar{e} \\ & + (\alpha_6\beta_5 - \alpha_5\beta_6)e_1e\bar{e} + (\alpha_4\beta_6 - \alpha_6\beta_4)e_2e\bar{e} + (\alpha_5\beta_4 - \alpha_4\beta_5)e_3e\bar{e} \\ & + (\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1)e \\ & + (\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1)\bar{e} \end{aligned} \tag{A.48}$$

De este resultado el término que indica si hay intersección es,

$$(\alpha_1\beta_6 + \alpha_2\beta_4 + \alpha_3\beta_5 + \alpha_4\beta_2 + \alpha_5\beta_3 + \alpha_6\beta_1) \tag{A.49}$$

2.2 Recta-Plano

A continuación se procede a calcular la siguiente expresión:

$$B = (\Pi \vee L) = (I\Pi) \cdot L \tag{A.50}$$

De las ecuaciones (A.16) y (A.30) se tiene,

$$\begin{aligned} L =& \beta_1 e_1 e_2 e + \beta_1 e_1 e_2 \bar{e} + \beta_2 e_2 e_3 e + \beta_2 e_2 e_3 \bar{e} + \beta_3 e_3 e_1 e + \beta_3 e_3 e_1 \bar{e} \\ & - \beta_4 e_1 e \bar{e} - \beta_5 e_2 e \bar{e} - \beta_6 e_3 e \bar{e} \end{aligned} \tag{A.51}$$

$$\Pi = \omega_1 e_1 e_2 e_3 e + \omega_1 e_1 e_2 e_3 \bar{e} - \omega_2 e_1 e_2 e \bar{e} - \omega_3 e_2 e_3 e \bar{e} - \omega_2 e_3 e_1 e \bar{e} \tag{A.52}$$

Se calcula $I\Pi$,

$$\begin{aligned} I\Pi =& \omega_1 e_1 e_2 e_3 e \bar{e} e_1 e_2 e_3 e + \omega_1 e_1 e_2 e_3 e \bar{e} e_1 e_2 e_3 \bar{e} \\ & - \omega_2 e_1 e_2 e_3 e \bar{e} e_1 e_2 e \bar{e} - \omega_3 e_1 e_2 e_3 e \bar{e} e_2 e_3 e \bar{e} \\ & - \omega_2 e_1 e_2 e_3 e \bar{e} e_3 e_1 e \bar{e} \end{aligned} \tag{A.53}$$

De donde,

$$I\Pi = \omega_3 e_1 + \omega_4 e_2 + \omega_2 e_3 + \omega_1 e + \omega_1 \bar{e} \tag{A.54}$$

Calculando $B = I\Pi \cdot L$,

$$\begin{aligned} B =& \omega_3 \beta_1 e_2 e + \omega_3 \beta_1 e_2 \bar{e} - \omega_3 \beta_3 e_3 e - \omega_3 \beta_3 e_3 \bar{e} - \omega_3 \beta_4 e \bar{e} - \omega_4 \beta_1 e_1 e - \omega_4 \beta_1 e_1 \bar{e} \\ & + \omega_4 \beta_2 e_3 e + \omega_4 \beta_2 e_3 \bar{e} - \omega_4 \beta_5 e \bar{e} - \omega_2 \beta_2 e_2 e - \omega_2 \beta_2 e_2 \bar{e} + \omega_3 \beta_3 e_1 e + \omega_3 \beta_4 e_1 \bar{e} \\ & - \omega_2 \beta_6 e \bar{e} + \omega_1 \beta_1 e_1 e_2 + \omega_1 \beta_2 e_2 e_3 + \omega_1 \beta_3 e_3 e_1 + \omega_1 \beta_4 e_1 \bar{e} + \omega_1 \beta_5 e_2 \bar{e} + \omega_1 \beta_6 e_3 \bar{e} \\ & - \omega_1 \beta_1 e_1 e_2 - \omega_1 \beta_2 e_2 e_3 - \omega_1 \beta_3 e_3 e_1 + \omega_1 \beta_4 e_1 e + \omega_1 \beta_5 e_2 e + \omega_1 \beta_6 e_3 e \end{aligned} \tag{A.55}$$

Simplificando,

$$\begin{aligned} B =& (\omega_2 \beta_3 + \omega_1 \beta_4 - \omega_4 \beta_1) e_1 e + (\omega_2 \beta_3 + \omega_1 \beta_4 - \omega_4 \beta_1) e_1 \bar{e} \\ & (\omega_3 \beta_1 - \omega_2 \beta_2 + \omega_1 \beta_5) e_2 e + (\omega_3 \beta_1 - \omega_2 \beta_2 - \omega_1 \beta_5) e_2 \bar{e} \\ & (-\omega_3 \beta_3 + \omega_4 \beta_2 + \omega_1 \beta_6) e_3 e + (-\omega_3 \beta_3 + \omega_4 \beta_2 + \omega_1 \beta_6) e_3 \bar{e} \\ & (-\omega_3 \beta_4 - \omega_4 \beta_5 - \omega_2 \beta_6) e \bar{e} \end{aligned} \tag{A.56}$$

Calculando B^2 se obtiene,

$$B^2 = (\omega_3 \beta_4 + \omega_4 \beta_5 + \omega_2 \beta_6)^2 \tag{A.57}$$

2.3 Recta-Esfera

A continuación se procede a calcular la siguiente expresión:

$$B = (E \vee L) = (IE) \cdot L \tag{A.58}$$

De las ecuaciones (A.16) y (A.39) se obtiene,

$$\begin{aligned} L =& \beta_1 e_1 e_2 e + \beta_1 e_1 e_2 \bar{e} + \beta_2 e_2 e_3 e + \beta_2 e_2 e_3 \bar{e} + \beta_3 e_3 e_1 e + \beta_3 e_3 e_1 \bar{e} \\ & - \beta_4 e_1 e \bar{e} - \beta_5 e_2 e \bar{e} - \beta_6 e_3 e \bar{e} \end{aligned} \tag{A.59}$$

$$E = \mu_1 e_1 e_2 e_3 e + \mu_2 e_1 e_2 e_3 \bar{e} - \mu_3 e_1 e_2 e \bar{e} - \mu_4 e_2 e_3 e \bar{e} - \mu_5 e_3 e_1 e \bar{e} \tag{A.60}$$

Se calcula IE,

$$\begin{aligned} IE =& \mu_1 e_1 e_2 e_3 e \bar{e} e_1 e_2 e_3 e + \mu_2 e_1 e_2 e_3 e \bar{e} e_1 e_2 e_3 \bar{e} - \mu_3 e_1 e_2 e_3 e \bar{e} e_1 e_2 e \bar{e} \\ & - \mu_4 e_1 e_2 e_3 e \bar{e} e_2 e_3 e \bar{e} - \mu_5 e_1 e_2 e_3 e \bar{e} e_3 e_1 e \bar{e} \end{aligned} \tag{A.61}$$

De donde,

$$IE = \quad \mu_4 e_1 + \mu_5 e_2 + \mu_3 e_3 + \mu_2 e + \mu_1 \bar{e} \tag{A.62}$$

Ahora se calcula el producto interno $B = IE \cdot L$,

$$\begin{aligned} B =& \mu_4 \beta_1 e_2 e + \mu_4 \beta_1 e_2 \bar{e} - \mu_4 \beta_3 e_3 e - \mu_4 \beta_3 e_3 \bar{e} - \mu_4 \beta_3 e \bar{e} - \mu_5 \beta_1 e_1 e - \mu_5 \beta_1 e_1 \bar{e} \\ & + \mu_5 \beta_2 e_3 e + \mu_5 \beta_2 e_3 \bar{e} - \mu_5 \beta_5 e \bar{e} - \mu_3 \beta_1 e_2 e - \mu_3 \beta_2 e_2 \bar{e} + \mu_3 \beta_3 e_1 e + \mu_3 \beta_3 e_1 \bar{e} \\ & - \mu_3 \beta_6 e \bar{e} + \mu_2 \beta_1 e_1 e_2 + \mu_2 \beta_2 e_2 e_3 + \mu_2 \beta_3 e_3 e_1 + \mu_2 \beta_4 e_1 \bar{e} + \mu_2 \beta_5 e_2 \bar{e} + \mu_2 \beta_6 e_3 \bar{e} \\ & - \mu_1 \beta_1 e_1 e_2 - \mu_1 \beta_2 e_2 e_3 - \mu_1 \beta_3 e_3 e_1 + \mu_1 \beta_4 e_1 e + \mu_1 \beta_5 e_2 e + \mu_1 \beta_6 e_3 e \end{aligned} \tag{A.63}$$

Simplificando y agrupando,

$$\begin{aligned} B =& (\mu_3 \beta_3 + \mu_1 \beta_4 - \mu_5 \beta_1) e_1 e + (\mu_2 \beta_4 + \mu_3 \beta_3 - \mu_5 \beta_1) e_1 \bar{e} \\ & + (\mu_4 \beta_1 + \mu_1 \beta_5 - \mu_3 \beta_2) e_2 e + (\mu_4 \beta_1 + \mu_2 \beta_5 - \mu_3 \beta_2) e_2 \bar{e} \\ & + (\mu_5 \beta_2 + \mu_1 \beta_6 - \mu_4 \beta_3) e_3 e + (\mu_5 \beta_2 + \mu_2 \beta_6 - \mu_4 \beta_3) e_3 \bar{e} \\ & + (\mu_2 \beta_1 - \mu_1 \beta_1) e_1 e_2 + (\mu_2 \beta_2 - \mu_1 \beta_2) e_2 e_3 + (\mu_2 \beta_3 - \mu_1 \beta_3) e_3 e_1 \\ & - (\mu_4 \beta_4 + \mu_5 \beta_5 + \mu_3 \beta_6) e \bar{e} \end{aligned} \tag{A.64}$$

Ahora se calcula B^2, y se obtiene,

$$\begin{aligned} B^2 =& - (\mu_3 \beta_3 + \mu_1 \beta_4 - \mu_5 \beta_1)^2 + (\mu_2 \beta_4 + \mu_3 \beta_3 - \mu_5 \beta_1)^2 \\ & - (\mu_4 \beta_1 + \mu_1 \beta_5 - \mu_3 \beta_2)^2 + (\mu_4 \beta_1 + \mu_2 \beta_5 - \mu_3 \beta_2)^2 \\ & + (\mu_5 \beta_2 + \mu_1 \beta_6 - \mu_4 \beta_3)^2 + (\mu_5 \beta_2 + \mu_2 \beta_6 - \mu_4 \beta_3)^2 \\ & - (\mu_2 \beta_1 - \mu_1 \beta_1)^2 - (\mu_2 \beta_2 - \mu_1 \beta_2)^2 - (\mu_2 \beta_3 - \mu_1 \beta_3)^2 \\ & + (\mu_4 \beta_4 + \mu_5 \beta_5 + \mu_3 \beta_6)^2 \end{aligned} \tag{A.65}$$

2.4 Plano-Plano

A continuación se procede a calcular la siguiente expresión:

$$B = (\Pi_1 \vee \Pi_2) = (I\Pi_1) \cdot \Pi_2 \tag{A.66}$$

De las ecuaciones (A.53) y (A.30) se tiene,

$$I\Pi_1 = \omega_3 e_1 + \omega_4 e_2 + \omega_2 e_3 + \omega_1 e + \omega_1 \bar{e} \tag{A.67}$$

$$\Pi_2 = \lambda_1 e_1 e_2 e_3 e + \lambda_1 e_1 e_2 e_3 \bar{e} - \lambda_2 e_1 e_2 e \bar{e} - \lambda_3 e_2 e_3 e \bar{e} - \lambda_2 e_3 e_1 e \bar{e} \tag{A.68}$$

Calculando el producto interno $B = (I\Pi_1) \cdot \Pi_2$ se tiene,

$$\begin{aligned} B = &\omega_3\lambda_1 e_2 e_3 e + \omega_3\lambda_1 e_2 e_3 \bar{e} - \omega_3\lambda_2 e_2 e \bar{e} + \omega_3\lambda_4 e_3 e \bar{e} - \omega_4\lambda_1 e_1 e_3 e - \omega_4\lambda_1 e_1 e_3 \bar{e} \\ &+ \omega_4\lambda_2 e_1 e \bar{e} - \omega_4\lambda_3 e_3 e \bar{e} + \omega_2\lambda_1 e_1 e_2 e + \omega_2\lambda_1 e_1 e_2 \bar{e} + \omega_2\lambda_3 e_2 e \bar{e} - \omega_2\lambda_4 e_1 e \bar{e} \\ &- \omega_1\lambda_1 e_1 e_2 e_3 - \omega_1\lambda_2 e_1 e_2 \bar{e} - \omega_1\lambda_3 e_2 e_3 \bar{e} - \omega_1\lambda_4 e_3 e_1 \bar{e} + \omega_1\lambda_1 e_1 e_2 e_3 - \omega_1\lambda_2 e_1 e_2 e \\ &- \omega_1\lambda_3 e_2 e_3 e - \omega_1\lambda_4 e_3 e_1 e \end{aligned} \tag{A.69}$$

Simplificando,

$$\begin{aligned} B = &(\omega_4\lambda_2 - \omega_2\lambda_4) e_1 e \bar{e} + (\omega_2\lambda_3 - \omega_3\lambda_2) e_2 e \bar{e} + (\omega_3\lambda_4 - \omega_4\lambda_3) e_3 e \bar{e} \\ &(\omega_2\lambda_1 - \omega_1\lambda_2) e_1 e_2 e + (\omega_3\lambda_1 - \omega_1\lambda_3) e_2 e_3 e + (\omega_4\lambda_1 - \omega_1\lambda_4) e_3 e_1 e \\ &(\omega_2\lambda_1 - \omega_1\lambda_2) e_1 e_2 \bar{e} + (\omega_3\lambda_1 - \omega_1\lambda_3) e_2 e_3 \bar{e} + (\omega_4\lambda_1 - \omega_1\lambda_4) e_3 e_1 \bar{e} \end{aligned} \tag{A.70}$$

Ahora se calcula B^2,

$$B^2 = (\omega_4\lambda_2 - \omega_2\lambda_4)^2 + (\omega_2\lambda_3 - \omega_3\lambda_2)^2 + (\omega_3\lambda_4 - \omega_4\lambda_3)^2 \tag{A.71}$$

2.5 Esfera-Plano

A continuación se procede a calcular la siguiente expresión:

$$B = (E \vee \Pi) = (IE) \cdot \Pi \tag{A.72}$$

De las ecuaciones (A.62) y (A.30) se obtiene,

$$IE = \mu_4 e_1 + \mu_5 e_2 + \mu_3 e_3 + \mu_2 e + \mu_1 \bar{e} \tag{A.73}$$

$$\Pi = \omega_1 e_1 e_2 e_3 e + \omega_1 e_1 e_2 e_3 \bar{e} - \omega_2 e_1 e_2 e \bar{e} - \omega_3 e_2 e_3 e \bar{e} - \omega_2 e_3 e_1 e \bar{e} \tag{A.74}$$

Calculando el producto interno $B = (IE) \cdot \Pi$ se tiene,

$$\begin{aligned} B = &\mu_4\omega_1 e_2 e_3 e + \mu_4\omega_1 e_2 e_3 \bar{e} - \mu_4\omega_2 e_2 e \bar{e} + \mu_4\omega_4 e_3 e \bar{e} - \mu_5\omega_1 e_1 e_3 e - \mu_5\omega_1 e_1 e_3 \bar{e} \\ &+ \mu_5\omega_2 e_1 e \bar{e} - \mu_5\omega_3 e_3 e \bar{e} + \mu_3\omega_1 e_1 e_2 e + \mu_3\omega_1 e_1 e_2 \bar{e} + \mu_3\omega_3 e_2 e \bar{e} - \mu_3\omega_4 e_1 e \bar{e} \\ &- \mu_2\omega_1 e_1 e_2 e_3 - \mu_2\omega_2 e_1 e_2 \bar{e} - \mu_2\omega_3 e_2 e_3 \bar{e} - \mu_2\omega_4 e_3 e_1 \bar{e} + \mu_1\omega_1 e_1 e_2 e_3 - \mu_1\omega_2 e_1 e_2 e \\ &- \mu_1\omega_3 e_2 e_3 e - \mu_1\omega_4 e_3 e_1 e \end{aligned} \tag{A.75}$$

Simplificando se obtiene,

$$\begin{aligned} B = &(\mu_5\omega_2 - \mu_3\omega_4) e_1 e \bar{e} + (\mu_3\omega_3 - \mu_4\omega_2) e_2 e \bar{e} + (\mu_4\omega_4 - \mu_5\omega_3) e_3 e \bar{e} \\ &+ (\mu_3\omega_1 - \mu_1\omega_2) e_1 e_2 e + (\mu_4\omega_1 - \mu_1\omega_3) e_2 e_3 e + (\mu_5\omega_1 - \mu_1\omega_4) e_3 e_1 e \\ &+ (\mu_3\omega_1 - \mu_2\omega_2) e_1 e_2 \bar{e} + (\mu_4\omega_1 - \mu_2\omega_3) e_2 e_3 \bar{e} + (\mu_5\omega_1 - \mu_2\omega_4) e_3 e_1 \bar{e} \\ &+ (\mu_1\omega_1 - \mu_2\omega_1) e_1 e_2 e_3 \end{aligned} \tag{A.76}$$

Se calcula B^2,

$$\begin{aligned} B^2 = &(\mu_5\omega_2 - \mu_3\omega_4)^2 + (\mu_3\omega_3 - \mu_4\omega_2)^2 + (\mu_4\omega_4 - \mu_5\omega_3)^2 \\ &- (\mu_3\omega_1 - \mu_1\omega_2)^2 - (\mu_4\omega_1 - \mu_1\omega_3)^2 - (\mu_5\omega_1 - \mu_1\omega_4)^2 \\ &+ (\mu_3\omega_1 - \mu_2\omega_2)^2 + (\mu_4\omega_1 - \mu_2\omega_3)^2 + (\mu_5\omega_1 - \mu_2\omega_4)^2 - (\mu_1\omega_1 - \mu_2\omega_1)^2 \end{aligned} \tag{A.77}$$

2.6 Esfera-Esfera

A continuación se procede a calcular la siguiente expresión:

$$B = (E_1 \vee E_2) = (IE_1) \cdot E_2 \tag{A.78}$$

De las ecuaciones (A.39) y (A.62) se tiene,

$$E_2 = \mu_1 e_1 e_2 e_3 e + \mu_2 e_1 e_2 e_3 \bar{e} - \mu_3 e_1 e_2 e \bar{e} - \mu_4 e_2 e_3 e \bar{e} - \mu_5 e_3 e_1 e \bar{e} \tag{A.79}$$

$$IE_1 = \quad \lambda_4 e_1 + \lambda_5 e_2 + \lambda_3 e_3 + \lambda_2 e + \lambda_1 \bar{e} \tag{A.80}$$

Ahora se muestra el producto interno $B = IE_1 \cdot E_2$,

$$\begin{aligned} B = &\mu_4\lambda_1 e_2 e_3 e + \mu_4\lambda_2 e_2 e_3 \bar{e} - \mu_4\lambda_3 e_2 e \bar{e} + \mu_4\lambda_5 e_3 e \bar{e} - \mu_5\lambda_1 e_1 e_3 \bar{e} - \mu_5\lambda_2 e_1 e_3 \bar{e} \\ &+ \mu_5\lambda_3 e_1 e \bar{e} - \mu_5\lambda_4 e_3 e \bar{e} + \mu_3\lambda_1 e_1 e_2 e + \mu_3\lambda_2 e_1 e_2 \bar{e} + \mu_3\lambda_4 e_2 e \bar{e} - \mu_3\lambda_5 e_1 e \bar{e} \\ &- \mu_2\lambda_1 e_1 e_2 e_3 - \mu_2\lambda_3 e_1 e_2 \bar{e} - \mu_2\lambda_4 e_2 e_3 \bar{e} - \mu_2\lambda_5 e_3 e_1 \bar{e} + \mu_1\lambda_2 e_1 e_2 e_3 \\ &- \mu_1\lambda_3 e_1 e_2 e - \mu_1\lambda_4 e_2 e_3 e - \mu_1\lambda_5 e_3 e_1 e \end{aligned} \tag{A.81}$$

Simplificando se obtiene,

$$\begin{aligned} B =&(\mu_3\lambda_1 - \mu_1\lambda_3)e_1e_2e + (\mu_4\lambda_1 - \mu_1\lambda_4)e_2e_3e + (\mu_5\lambda_1 - \mu_1\lambda_5)e_3e_1e \\ &+ (\mu_3\lambda_2 - \mu_2\lambda_3)e_1e_2\bar{e} + (\mu_4\lambda_2 - \mu_2\lambda_4)e_2e_3\bar{e} + (\mu_5\lambda_2 - \mu_2\lambda_5)e_3e_1\bar{e} \\ &+ (\mu_5\lambda_3 - \mu_3\lambda_5)e_1e\bar{e} + (\mu_3\lambda_4 - \mu_4\lambda_3)e_2e\bar{e} + (\mu_4\lambda_5 - \mu_5\lambda_4)e_3e\bar{e} \\ &+ (\mu_1\lambda_2 - \mu_2\lambda_1)e_1e_2e_3 \end{aligned} \tag{A.82}$$

Calculando B^2 se obtiene,

$$\begin{aligned} B^2 =& -(\mu_3\lambda_1 - \mu_1\lambda_3)^2 - (\mu_4\lambda_1 - \mu_1\lambda_4)^2 - (\mu_5\lambda_1 - \mu_1\lambda_5)^2 \\ &+ (\mu_3\lambda_2 - \mu_2\lambda_3)^2 + (\mu_4\lambda_2 - \mu_2\lambda_4)^2 + (\mu_5\lambda_2 - \mu_2\lambda_5)^2 \\ &+ (\mu_5\lambda_3 - \mu_3\lambda_5)^2 + (\mu_3\lambda_4 - \mu_4\lambda_3)^2 + (\mu_4\lambda_5 - \mu_5\lambda_4)^2 \\ &- (\mu_1\lambda_2 - \mu_2\lambda_1)^2 \end{aligned} \tag{A.83}$$

3 Rotación

Del capítulo II tenemos la siguiente expresión para rotar un punto alrededor del origen,

$$X' = RX\tilde{R} \tag{A.84}$$

Donde R y $\tilde{R}$ se definen como,

$$\begin{aligned} R &= cos(\alpha) - BSen(\alpha) \\ \tilde{R} &= cos(\alpha) + BSen(\alpha) \end{aligned} \tag{A.85}$$

Y donde X es igual,

$$X = 2x + x^2n - \bar{n} \tag{A.86}$$

$$x = a_1e_1 + a_2e_2 + a_3e_3 \tag{A.87}$$

Para resolver los problemas planteados en la tesis, solo se necesita rotar en los 3 ejes principales x, y, z, en este caso se muestra el cálculo para rotar alrededor de y, es decir, $B = e_3 \wedge e_1 = e_3e_1$. Tomando la ecuación (A.84) se tiene,

$$\begin{aligned} X' &= (cos(\alpha) - BSen(\alpha))X(cos(\alpha) + Bsen(\alpha)) \\ &= (cos(\alpha)X - BSen(\alpha)X)(cos(\alpha) + Bsen(\alpha)) \\ &= Xcos^2(\alpha) - BXBsen^2(\alpha) + (XB - BX)sen(\alpha)cos(\alpha) \end{aligned} \tag{A.88}$$

Para simplificar el cálculo, se calcula por separado cada término, empezando por $Xcos^2(\alpha)$,

$$Xcos^2(\alpha) = 2xcos^2(\alpha) + x^2ncos^2(\alpha) - \bar{n}cos^2(\alpha) \tag{A.89}$$

Ahora se calcula $-BXBsen^2(\alpha)$,

$$-BXBsen^2(\alpha) = -2BxBsen^2(\alpha) - BnBx^2sen^2(\alpha) + B\bar{n}Bsen^2(\alpha) \tag{A.90}$$

Ahora, $B = e_3e_1$, por lo tanto,

$$\begin{aligned}
-BXBsen^2(\alpha) &= -2e_3e_1xe_3e_1sen^2(\alpha) - e_3e_1(e+\bar{e})e_3e_1x^2sen^2(\alpha) \\
&+ e_3e_1(e-\bar{e})e_3e_1sen^2(\alpha) \\
&= -2e_3e_1xe_3e_1sen^2(\alpha) - (e_3e_1ee_3e_1 + e_3e_1\bar{e}e_3e_1)x^2sen^2(\alpha) \\
&+ (e_3e_1ee_3e_1 - e_3e_1\bar{e}e_3e_1)sen^2(\alpha) \\
&= -2e_3e_1xe_3e_1sen^2(\alpha) - (e_3e_1e_3e_1e + e_3e_1e_3e_1\bar{e})x^2sen^2(\alpha) \\
&+ (e_3e_1e_3e_1e - e_3e_1e_3e_1\bar{e})sen^2(\alpha) \\
&= -2e_3e_1xe_3e_1sen^2(\alpha) - (-e-\bar{e})x^2sen^2(\alpha) \\
&+ (-e+\bar{e})sen^2(\alpha) \\
-BXBsen^2(\alpha) &= -2e_3e_1xe_3e_1sen^2(\alpha) + nx^2sen^2(\alpha) - \bar{n}sen^2(\alpha)
\end{aligned} \tag{A.91}$$

Ahora se calcula $(XB - BX)sen(\alpha)cos(\alpha)$,

$$\begin{aligned}
(XB - BX) &= (2x + x^2n - \bar{n})e_3e_1 - e_3e_1(2x + x^2n - \bar{n}) \\
&= 2xe_3e_1 + x^2ne_3e_1 - \bar{n}e_3e_1 - 2e_3e_1x - x^2e_3e_1n + e_3e_1\bar{n} \\
&= 2xe_3e_1 + x^2(ee_3e_1 + \bar{3}e_3e_1) + (ee_3e_1 - \bar{e}e_3e_1) - 2e_3e_1x \\
&- x^2(e_3e_1e + e_3e_1\bar{e}) + e_3e_1e - e_3e_1\bar{e} \\
&= (2xe_3e_1 - 2e_3e_1x)sen(\alpha)cos(\alpha)
\end{aligned} \tag{A.92}$$

Retornando a (A.88) se tiene,

$$\begin{aligned}
X' &= 2xcos^2(\alpha) + x^2ncos^2(\alpha) - \bar{n}cos^2e + \bar{e} - 2e_3e_1xe_3e_1sen^2(\alpha) \\
&+ nx^2sen^2(\alpha) - \bar{n}sen^2(\alpha) + (2xe_3e_1 - 2e_3e_1x)sen(\alpha)cos(\alpha)
\end{aligned} \tag{A.93}$$

Simplificando,

$$\begin{aligned}
X' &= 2[xcos^2(\alpha) - e_3e_1xe_3e_1sen^2(\alpha) + (xe_3e_1 - e_3e_1x)sen(\alpha)cos(\alpha)] \\
&+ x^2n - \bar{n}+
\end{aligned} \tag{A.94}$$

De esta última expresión se tiene, que el punto rotado se obtiene al resolver $x' = [xcos^2(\alpha) - e_3e_1xe_3e_1sen^2(\alpha) + (xe_3e_1 - e_3e_1x)sen(\alpha)cos(\alpha)]$. El cálculo de esta expresión da como resultado,

$$\begin{aligned} x' = & e_1[a_1cos^2(\alpha) - a_1sen^2(\alpha) + 2a_3sen(\alpha)cos(\alpha)] \\ & + e_2 \\ & + e_3[a_3cos^2(\alpha) - a_3sen^2(\alpha) - 2a_1sen(\alpha)cos(\alpha)] \end{aligned} \tag{A.95}$$

Si se quiere rotar alrededor de x, es decir, $B = e_3 \wedge e_2$, se obtiene,

$$\begin{aligned} x' = & e_1 \\ & + e_2[a_2cos^2(\alpha) - a_2sen^2(\alpha) + 2a_3sen(\alpha)cos(\alpha)] \\ & + e_3[a_3cos^2(\alpha) - a_3sen^2(\alpha) - 2a_2sen(\alpha)cos(\alpha)] \end{aligned} \tag{A.96}$$

Y por último rotando alrededor de z, es decir, $B = e_1 \wedge e_2$, se obtiene,

$$\begin{aligned} x' = & e_1[a_1cos^2(\alpha) - a_1sen^2(\alpha) + 2a_2sen(\alpha)cos(\alpha)] \\ & + e_2[a_2cos^2(\alpha) - a_2sen^2(\alpha) - 2a_1sen(\alpha)cos(\alpha)] \\ & + e_3 \end{aligned} \tag{A.97}$$

4 Reflexión

A continuación se calcula la siguiente expresión:

$$L' = LPL \tag{A.98}$$

Donde L' es la recta L reflejada en el plano P.

$$\begin{aligned} L = 4[&\beta_1e_1e_2e + \beta_1e_1e_2\bar{e} + \beta_2e_2e_3e + \beta_2e_2e_3\bar{e} + \beta_3e_3e_1e + \beta_3e_3e_1\bar{e} \\ &- \beta_4e_1e\bar{e} - \beta_5e_2e\bar{e} - \beta_6e_3e\bar{e}] \end{aligned} \tag{A.99}$$

$$P = 8[\omega_1e_1e_2e_3e + \omega_1e_1e_2e_3\bar{e} - \omega_2e_1e_2e\bar{e} - \omega_3e_2e_3e\bar{e} - \omega_2e_3e_1e\bar{e}] \tag{A.100}$$

Se calcula primero PL,

$$
\begin{aligned}
PL =& \omega_1\beta_1 e_1e_2e_3ee_1e_2e + \omega_1\beta_1 e_1e_2e_3ee_1e_2\bar{e} + \omega_1\beta_2 e_1e_2e_3ee_2e_3e \\
&+ \omega_1\beta_2 e_1e_2e_3ee_2e_3\bar{e} + \omega_1\beta_3 e_1e_2e_3ee_3e_1e + \omega_1\beta_3 e_1e_2e_3ee_3e_1\bar{e} \\
&- \omega_1\beta_4 e_1e_2e_3ee_1e\bar{e} - \omega_1\beta_5 e_1e_2e_3ee_2e\bar{e} - \omega_1\beta_6 e_1e_2e_3ee_3e\bar{e} \\
&+ \omega_1\beta_1 e_1e_2e_3\bar{e}e_1e_2e + \omega_1\beta_1 e_1e_2e_3\bar{e}e_1e_2\bar{e} + \omega_1\beta_2 e_1e_2e_3\bar{e}e_2e_3e \\
&+ \omega_1\beta_2 e_1e_2e_3\bar{e}e_2e_3\bar{e} + \omega_1\beta_3 e_1e_2e_3\bar{e}e_3e_1e + \omega_1\beta_3 e_1e_2e_3\bar{e}e_3e_1\bar{e} \\
&- \omega_1\beta_4 e_1e_2e_3\bar{e}e_1e\bar{e} - \omega_1\beta_5 e_1e_2e_3\bar{e}e_2e\bar{e} - \omega_1\beta_6 e_1e_2e_3\bar{e}e_3e\bar{e} \\
&- \omega_2\beta_1 e_1e_2e\bar{e}e_1e_2e - \omega_2\beta_1 e_1e_2e\bar{e}e_1e_2\bar{e} - \omega_2\beta_2 e_1e_2e\bar{e}e_2e_3e \\
&- \omega_2\beta_2 e_1e_2e\bar{e}e_2e_3\bar{e} - \omega_2\beta_3 e_1e_2e\bar{e}e_3e_1e - \omega_2\beta_3 e_1e_2e\bar{e}e_3e_1\bar{e} \\
&+ \omega_2\beta_4 e_1e_2e\bar{e}e_1e\bar{e} + \omega_2\beta_5 e_1e_2e\bar{e}e_2e\bar{e} + \omega_2\beta_6 e_1e_2e\bar{e}e_3e\bar{e} \\
&- \omega_3\beta_1 e_2e_3e\bar{e}e_1e_2e - \omega_3\beta_1 e_2e_3e\bar{e}e_1e_2\bar{e} - \omega_3\beta_2 e_2e_3e\bar{e}e_2e_3e \\
&- \omega_3\beta_2 e_2e_3e\bar{e}e_2e_3\bar{e} - \omega_3\beta_3 e_2e_3e\bar{e}e_3e_1e - \omega_3\beta_3 e_2e_3e\bar{e}e_3e_1\bar{e} \\
&+ \omega_3\beta_4 e_2e_3e\bar{e}e_1e\bar{e} + \omega_3\beta_5 e_2e_3e\bar{e}e_2e\bar{e} + \omega_3\beta_6 e_2e_3e\bar{e}e_3e\bar{e} \\
&- \omega_4\beta_1 e_3e_1e\bar{e}e_1e_2e - \omega_4\beta_1 e_3e_1e\bar{e}e_1e_2\bar{e} - \omega_4\beta_2 e_3e_1e\bar{e}e_2e_3e \\
&- \omega_4\beta_2 e_3e_1e\bar{e}e_2e_3\bar{e} - \omega_4\beta_3 e_3e_1e\bar{e}e_3e_1e - \omega_4\beta_3 e_3e_1e\bar{e}e_3e_1\bar{e} \\
&+ \omega_4\beta_4 e_3e_1e\bar{e}e_1e\bar{e} + \omega_4\beta_5 e_3e_1e\bar{e}e_2e\bar{e} + \omega_4\beta_6 e_3e_1e\bar{e}e_3e\bar{e}
\end{aligned}
\tag{A.101}
$$

$$
\begin{aligned}
PL =& -\omega_1\beta_1 e_3 - \omega_1\beta_1 e_3e\bar{e} - \omega_1\beta_2 e_1 - \omega_1\beta_2 e_1e\bar{e} - \omega_1\beta_3 e_2 - \omega_1\beta_3 e_2e\bar{e} \\
&+ \omega_1\beta_4 e_2e_3\bar{e} - \omega_1\beta_5 e_1e_3\bar{e} + \omega_1\beta_6 e_1e_2\bar{e} + \omega_1\beta_1 e_3e\bar{e} + \omega_1\beta_1 e_3 + \omega_1\beta_2 e_1e\bar{e} \\
&+ \omega_1\beta_2 e_1 + \omega_1\beta_3 e_2e\bar{e} + \omega_1\beta_3 e_2 + \omega_1\beta_4 e_2e_3e - \omega_1\beta_5 e_1e_3e + \omega_1\beta_6 e_1e_2e \\
&- \omega_2\beta_1\bar{e} - \omega_2\beta_1 e - \omega_2\beta_2 e_3e_1\bar{e} - \omega_2\beta_2 e_3e_1e + \omega_2\beta_3 e_2e_3\bar{e} - \omega_2\beta_3 e_2e_3e \\
&- \omega_2\beta_4 e_2 + \omega_2\beta_5 e_1 + \omega_2\beta_6 e_1e_2e_3 + \omega_3\beta_1 e_3e_1\bar{e} + \omega_3\beta_1 e_3e_1e - \omega_3\beta_2\bar{e} \\
&- \omega_3\beta_2 e - \omega_3\beta_3 e_1e_2\bar{e} - \omega_3\beta_3 e_1e_2e + \omega_3\beta_4 e_1e_2e_3 - \omega_3\beta_5 e_3 + \omega_3\beta_6 e_2 \\
&- \omega_4\beta_1 e_2e_3\bar{e} - \omega_4\beta_1 e_2e_3e + \omega_4\beta_2 e_1e_2\bar{e} + \omega_4\beta_2 e_1e_2e - \omega_4\beta_3\bar{e} - \omega_4\beta_3 e \\
&+ \omega_4\beta_4 e_3 + \omega_4\beta_5 e_1e_2e_3 - \omega_4\beta_6 e_1
\end{aligned}
\tag{A.102}
$$

Simplificando y factorizando,

$$
\begin{aligned}
PL =& (\omega_2\beta_5 - \omega_4\beta_6)e_1 + (\omega_3\beta_6 - \omega_2\beta_4)e_2 + (\omega_4\beta_4 - \omega_3\beta_5)e_3 \\
&- (\omega_2\beta_1 + \omega_3\beta_2 + \omega_4\beta_3)e - (\omega_2\beta_1 + \omega_3\beta_2 + \omega_4\beta_3)\bar{e} + (\omega_1\beta_6 - \omega_3\beta_3 + \omega_4\beta_2)e_1e_2e \\
&+ (\omega_1\beta_4 + \omega_2\beta_3 - \omega_4\beta_1)e_2e_3e + (\omega_1\beta_5 - \omega_2\beta_2 + \omega_3\beta_1)e_3e_1e \\
&+ (\omega_1\beta_6 - \omega_3\beta_3 + \omega_4\beta_2)e_1e_2\bar{e} + (\omega_1\beta_4 + \omega_2\beta_3 - \omega_4\beta_1)e_2e_3\bar{e} \\
&+ (\omega_1\beta_5 - \omega_2\beta_2 + \omega_3\beta_1)e_3e_1\bar{e} + (\omega_2\beta_6 + \omega_3\beta_4 + \omega_4\beta_5)e_1e_2e_3
\end{aligned}
\tag{A.103}
$$

Ahora se calcula $L' = PLP$,

$$
\begin{aligned}
L' =&\omega_1(\omega_2\beta_5-\omega_4\beta_6)e_2e_3e+\omega_1(\omega_2\beta_5-\omega_4\beta_6)e_2e_3\bar{e}-\omega_2(\omega_2\beta_5-\omega_4\beta_6)e_2e\bar{e}\\
&-\omega_3(\omega_2\beta_5-\omega_4\beta_6)e_1e_2e_3e\bar{e}+\omega_4(\omega_2\beta_5-\omega_4\beta_6)e_3e\bar{e}-\omega_1(\omega_3\beta_6-\omega_2\beta_4)e_1e_3e\\
&-\omega_1(\omega_3\beta_6-\omega_2\beta_4)e_1e_3\bar{e}+\omega_2(\omega_3\beta_6-\omega_2\beta_4)e_1e\bar{e}-\omega_3(\omega_3\beta_6-\omega_2\beta_4)e_3\bar{e}\\
&-\omega_4(\omega_3\beta_6-\omega_2\beta_4)e_1e_2e_3e\bar{e}+\omega_1(\omega_4\beta_4-\omega_3\beta_5)e_1e_2e+\omega_1(\omega_4\beta_4-\omega_3\beta_5)e_1e_2\bar{e}\\
&-\omega_2(\omega_4\beta_4-\omega_3\beta_5)e_1e_2e_3e\bar{e}+\omega_3(\omega_4\beta_4-\omega_3\beta_5)e_2e\bar{e}-\omega_4(\omega_4\beta_4-\omega_3\beta_5)e_1e\bar{e}\\
&\omega_1(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2e_3+\omega_1(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2e_3e\bar{e}\\
&+\omega_2(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2\bar{e}+\omega_3(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_2e_3\bar{e}\\
&+\omega_4(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_3e_1\bar{e}-\omega_1(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2e_3e\bar{e}\\
&-\omega_1(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2e_3+\omega_2(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_1e_2e\\
&+\omega_3(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_2e_3e+\omega_4(\omega_2\beta_1-\omega_3\beta_2+\omega_4\beta_3)e_3e_1e\\
&+\omega_1(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3+\omega_1(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3e\bar{e}\\
&+\omega_2(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)\bar{e}+\omega_3(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3e_1\bar{e}\\
&-\omega_4(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_2e_3\bar{e}+\omega_1(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_1\\
&+\omega_1(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_1e\bar{e}-\omega_2(\omega_1\beta_4+\omega_3\beta_3-\omega_4\beta_1)e_3e_1\bar{e}\\
&+\omega_3(\omega_1\beta_4+\omega_3\beta_3-\omega_4\beta_1)\bar{e}+\omega_4(\omega_1\beta_4+\omega_3\beta_3-\omega_4\beta_1)e_1e_2\bar{e}\\
&+\omega_1(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_2+\omega_1(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_2e\bar{e}\\
&-\omega_2(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_3e_2\bar{e}-\omega_3(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_1e_2\bar{e}\\
&+\omega_4(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)\bar{e}-\omega_1(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3e\bar{e}\\
&-\omega_1(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3+\omega_2(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e\\
&+\omega_3(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_3e_1e-\omega_4(\omega_1\beta_6-\omega_3\beta_3+\omega_4\beta_2)e_2e_3e\\
&-\omega_1(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_1e\bar{e}-\omega_1(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_1\\
&-\omega_2(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_3e_1e+\omega_3(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e\\
&+\omega_4(\omega_1\beta_4+\omega_2\beta_3-\omega_4\beta_1)e_1e_2e-\omega_1(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_2e\bar{e}\\
&-\omega_1(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_2+\omega_2(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_2e_3e\\
&-\omega_3(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e_1e_2e+\omega_1(\omega_1\beta_5-\omega_2\beta_2+\omega_3\beta_1)e\\
&-\omega_1(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)e-\omega_1(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)\bar{e}\\
&+\omega_2(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)e_3e\bar{e}+\omega_3(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)e_1e\bar{e}\\
&+\omega_4(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)e_2e\bar{e}
\end{aligned}
\tag{A.104}
$$

Anulando términos y simplificando se tiene,

$$
\begin{aligned}
L' =&[\omega_2(\omega_3\beta_6-\omega_2\beta_4)-\omega_4(\omega_4\beta_4-\omega_3\beta_5)+\omega_3(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)]e_1e\bar{e}\\
&+[-\omega_2(\omega_2\beta_5-\omega_4\beta_6)+\omega_3(\omega_4\beta_4-\omega_3\beta_5)+\omega_4(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)]e_2e\bar{e}\\
&+[\omega_4(\omega_2\beta_5-\omega_4\beta_6)-\omega_3(\omega_3\beta_6-\omega_2\beta_4)+\omega_2(\omega_2\beta_6+\omega_3\beta_4+\omega_4\beta_5)]e_3e\bar{e}
\end{aligned}
\tag{A.105}
$$

El vector reflejado L' en $\mathbb{R}^3$, es el vector,

$$
\begin{aligned}
L' =&[\omega_2(\omega_3\beta_6 - \omega_2\beta_4) - \omega_4(\omega_4\beta_4 - \omega_3\beta_5) + \omega_3(\omega_2\beta_6 + \omega_3\beta_4 + \omega_4\beta_5)]e_1 \\
&+ [-\omega_2(\omega_2\beta_5 - \omega_4\beta_6) + \omega_3(\omega_4\beta_4 - \omega_3\beta_5) + \omega_4(\omega_2\beta_6 + \omega_3\beta_4 + \omega_4\beta_5)]e_2 \\
&+ [\omega_4(\omega_2\beta_5 - \omega_4\beta_6) - \omega_3(\omega_3\beta_6 - \omega_2\beta_4) + \omega_2(\omega_2\beta_6 + \omega_3\beta_4 + \omega_4\beta_5)]e_3
\end{aligned} \tag{A.106}
$$

APÉNDICE B

CAPTURAS DE PANTALLA

En este apéndice se presentan unas capturas de pantalla de las intersecciones realizadas.

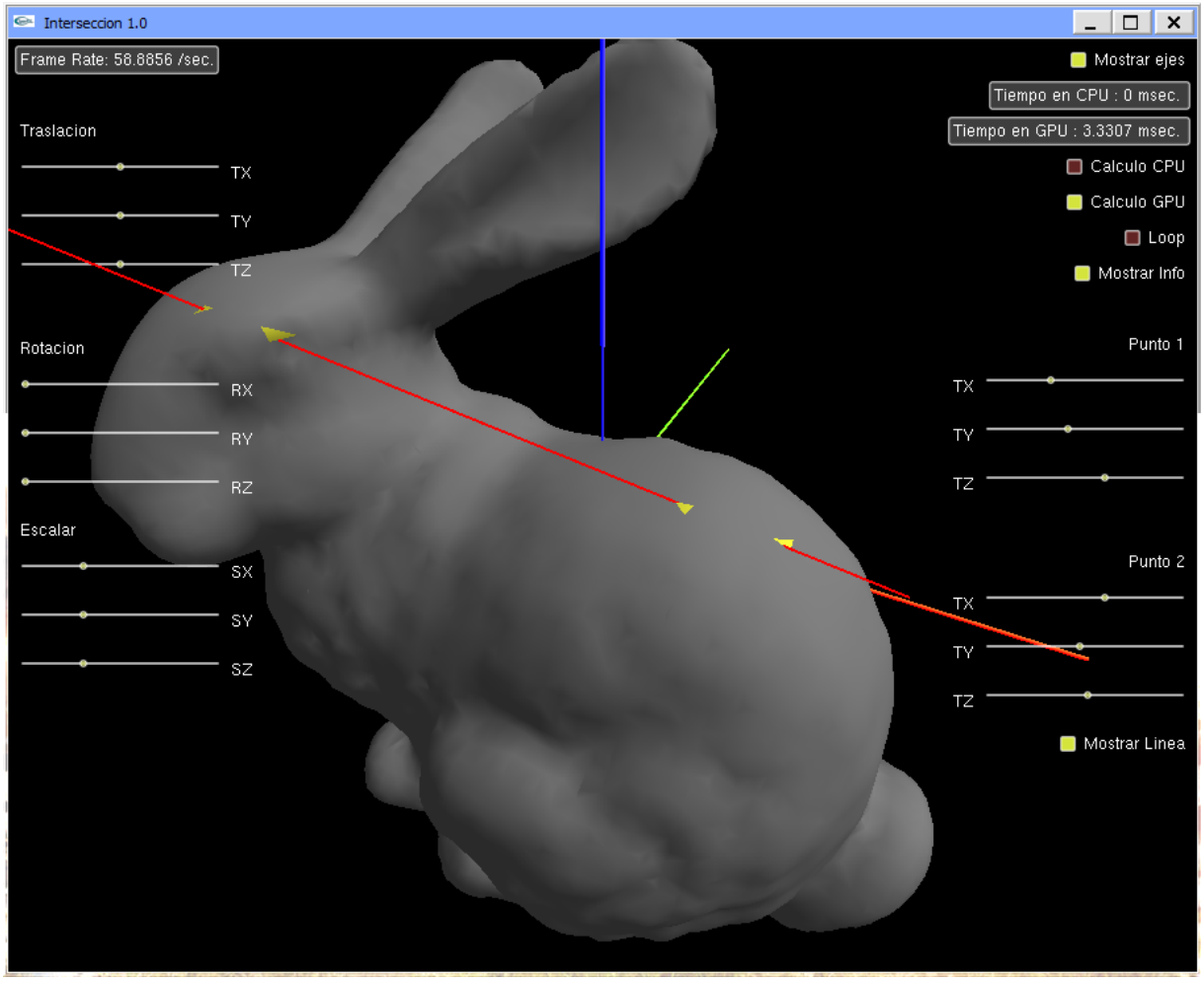


Figura B.1: *Intersección Malla Bunny - Segmento de Recta*

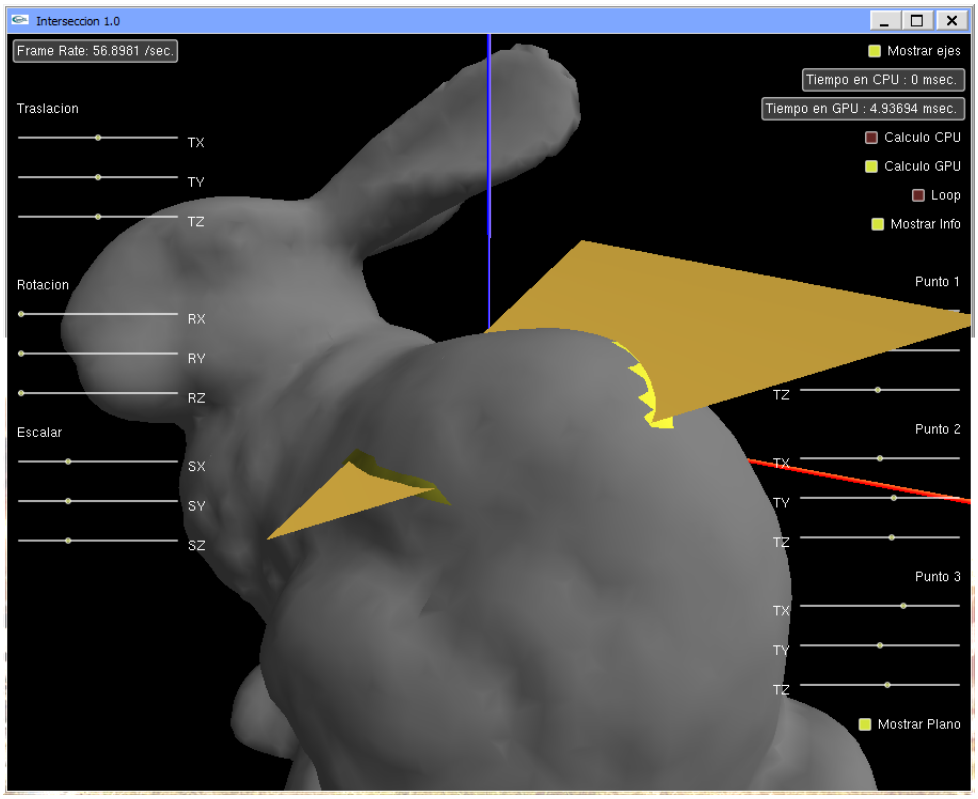


Figura B.2: *Intersección Malla Bunny - Triángulo*

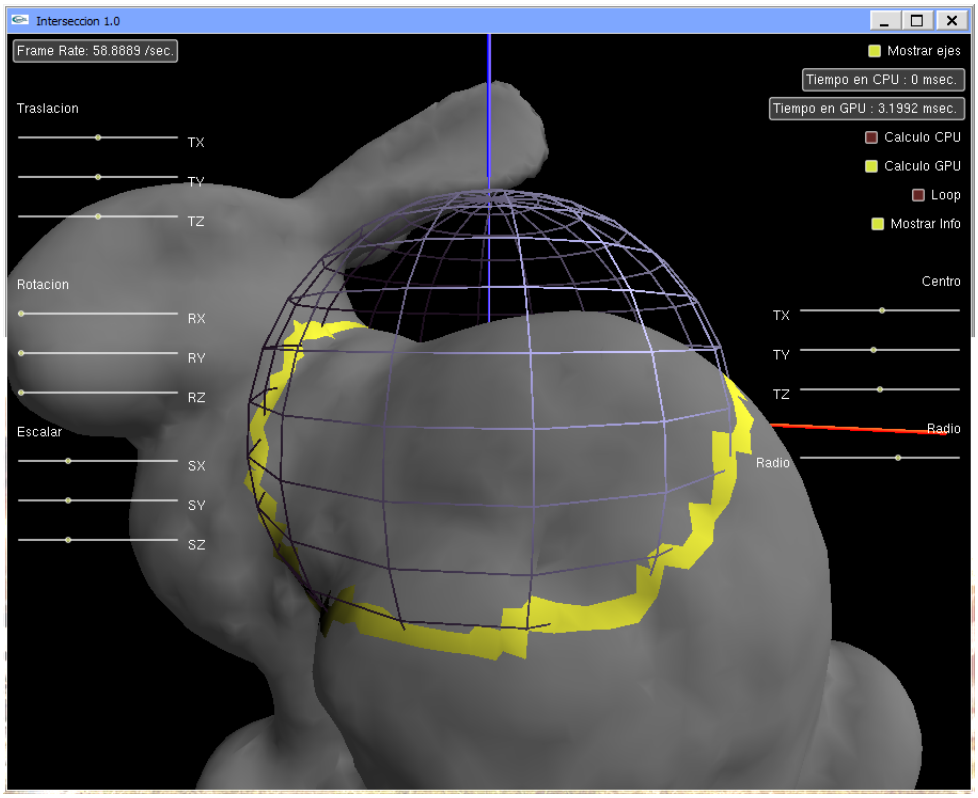


Figura B.3: *Intersección Malla Bunny - Esfera*

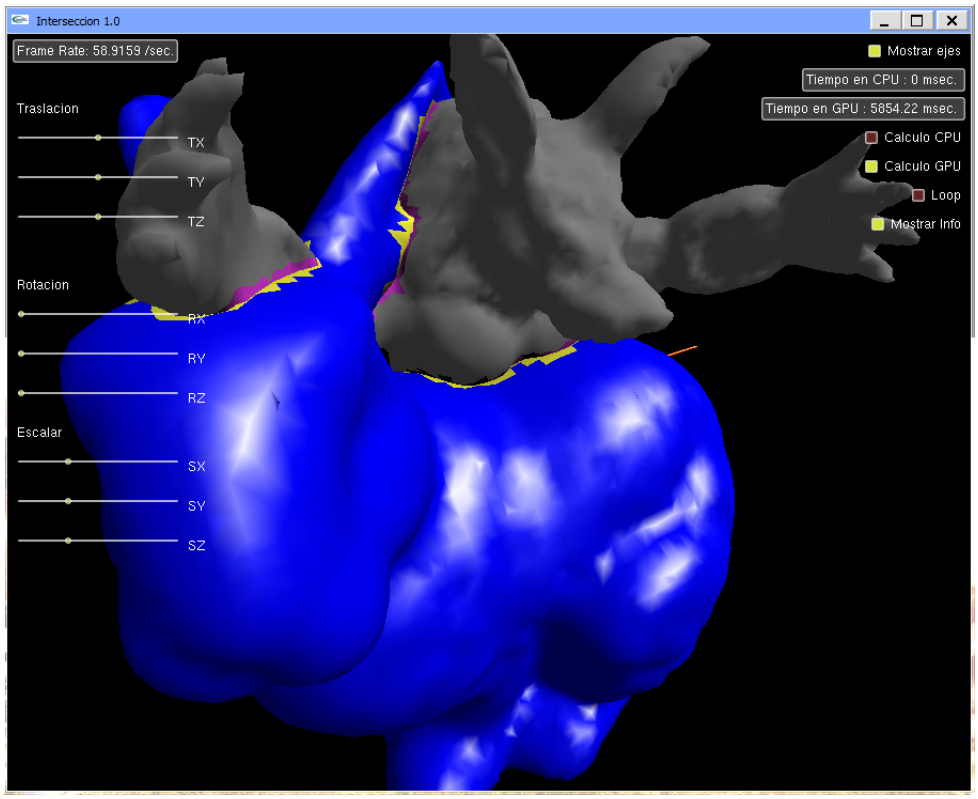


Figura B.4: *Intersección Malla Bunny - Malla Armadiillo*

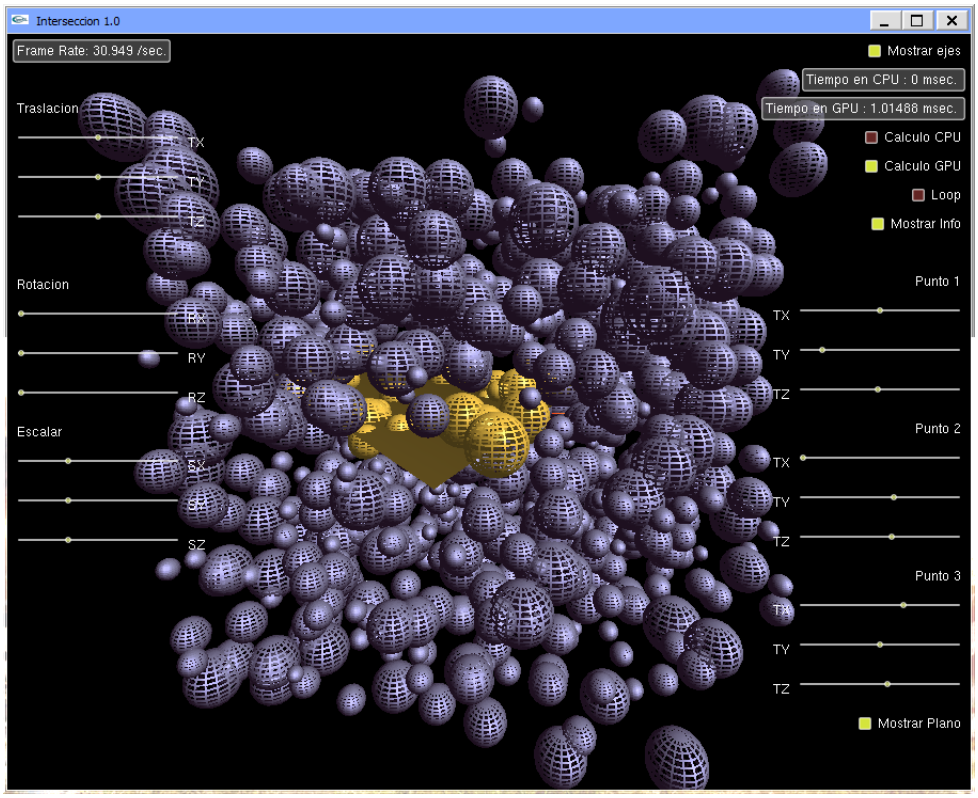


Figura B.5: *Intersección Esferas - Triángulo*

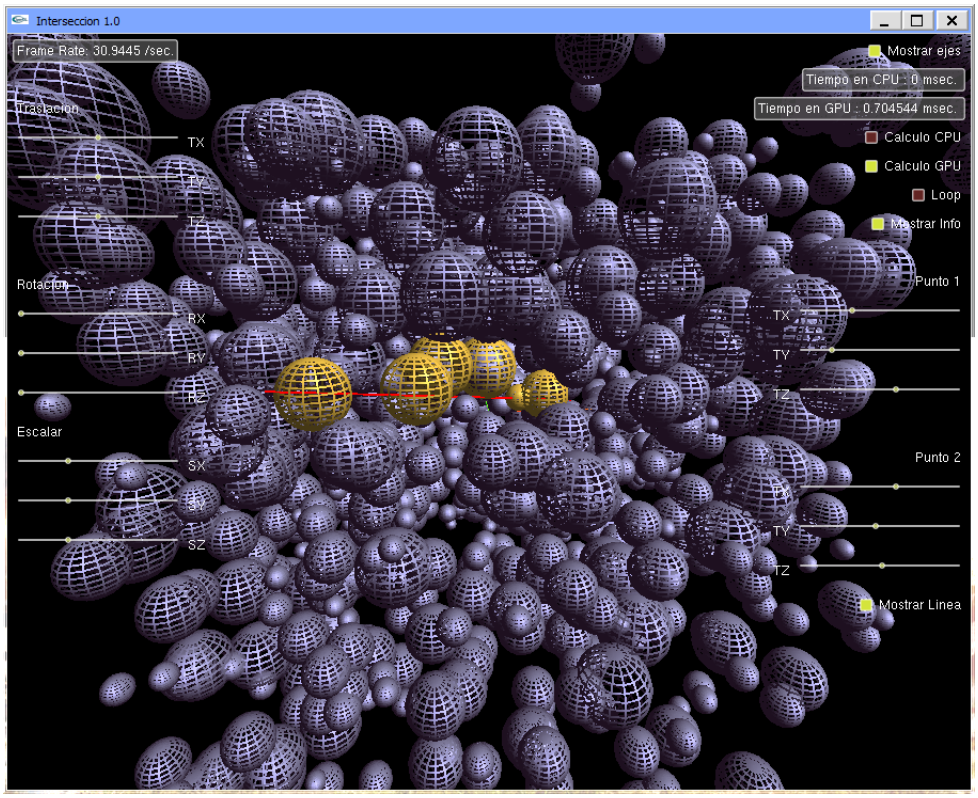


Figura B.6: *Intersección Esferas - Segmento de Recta*

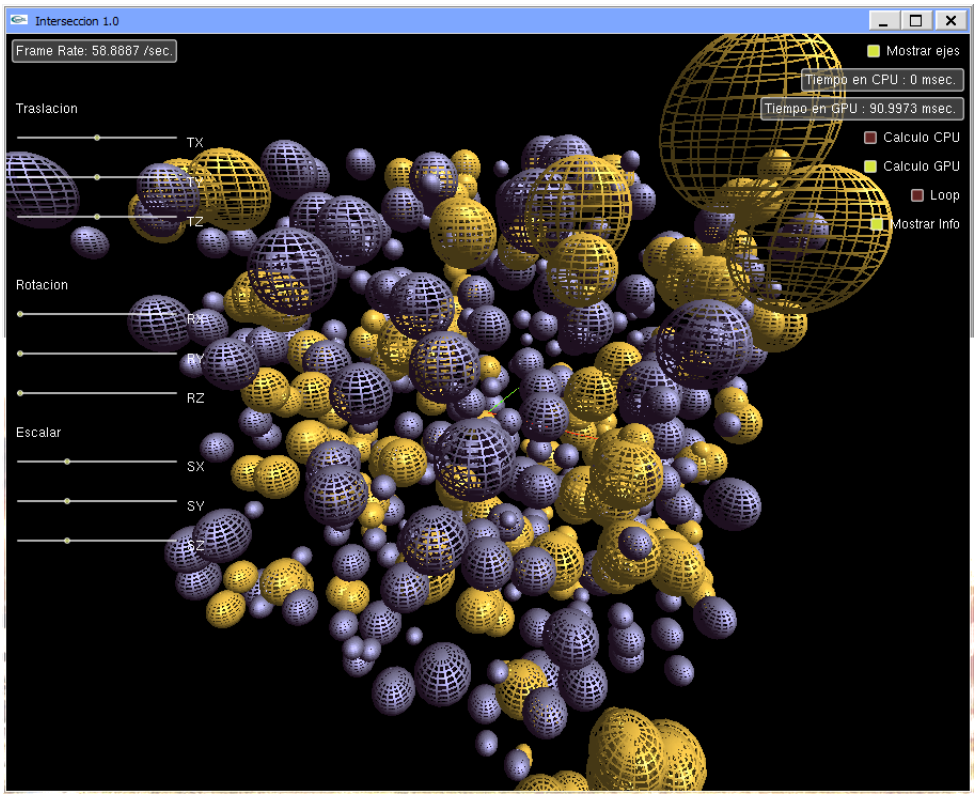


Figura B.7: *Intersección Esferas - Esferas*

REFERENCIAS BIBLIOGRÁFICAS

[1] Daniel Fontijne and Leo Dorst. Modeling 3d euclidean geometry. *IEEE Computer Graphics and Applications*, 23:68–78, 2003.

[2] J. Vince. *Geometric for Computer Graphics.* Springer, London, UK, 2005.

[3] J. Vince. *Geometric Algebra for Computer Graphics.* Springer, London, UK, 2008.

[4] A.J. Hanson. Visualizing quaternions. *Siggraph 2007*, Summer 2007.

[5] James R. Miller. Vector geometry for computer graphics. *IEEE Computer Graphics and Applications*, 19:66–73, 1999.

[6] Desmond Fearnley-sander. Hermann grassmann and the prehistory of universal algebra, 1982.

[7] D. Hestenes and G. Sobezyk. *Clifford Algebra to Geometric Calculus.* Reidel, Dordreeht, 1984.

[8] Zhigang Xiang and Roy Plastock. *Computer Graphics Second Edition, schaum's outlines.* McGraw-Hill, 2000.

[9] Ken Shoemake. Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, 19(3):245–254, 1985.

[10] James M. Van Verth and Lars M. Bishop. *Essential Mathematics for Games and Interactive Applications.* Morgan Kaufmann, San Francisco, 2004.

[11] `http://geometryalgebra.zcu.cz/`. Computer graphics and computer vision with geometric algebra and conformal geometry.

[12] C. Doran y A. Lasenby. *Geometric Algebra for physicist.* Cambridge, Cambridge, UK, 2009.

[13] D. Fontijne y S. Mann L. Dorst. *Geometric Algebra for Computer Science.* Morgan Kaufmann, San Francisco, CA, 2007.

[14] Chris Doran, Anthony N. Lasenby, and Joan Lasenby. Conformal geometry, euclidean space and geometric algebra. *CoRR*, cs.CG/0203026, 2002.

[15] `http://www.intel.com/technology/platform-technology/hyper-threading/index.htm`. Intel hyperthreading technology, 2010.

[16] `http://www.intel.com/info/hyperthreading/`. Intel hyper-threading technology, 2010.

[17] http://www.intel.com/espanol/technology/computing/dual-core/index.htm. Intel dual core, 2010.

[18] http://www.opengl.org/about/overview/. Opengl, 2010.

[19] http://www.futuretech.blinkenlights.nl/pcw9-93indy.html. Silicon graphics indy, 2010.

[20] http://en.wikipedia.org/wiki/S3Graphics. S3 graphics, ltd, 2010.

[21] http://en.wikipedia.org/wiki/3dfx. 3dfx interactive, 2010.

[22] http://www.nvidia.com/page/home.html. Nvidia corporation, 2010.

[23] http://msdn.microsoft.com/en-us/directx/default.aspx. Microsoft directx, 2010.

[24] http://en.wikipedia.org/wiki/SIMD. Single instruction, multiple data (simd), 2010.

[25] http://idsoftware.com/games/quake/quake/. Id software quake, 2010.

[26] http://developer.nvidia.com/node/76. The cg tutorial: Chapter 1. introduction, 2010.

[27] Greg Humphreys David Luebke. How gpu work. *IEEE Computer Graphics and Applications*, 40:96–100, 2007.

[28] http://developer.nvidia.com/page/home.html. Developer nvidia, 2010.

[29] http://udn.epicgames.com/Three/MaterialExamples.html. Bump mapping - unreal engine, 2010.

[30] http://http.developer.nvidia.com/GPUGems3/gpugems3_ch39.html. Depth of field - gpu gems 3, 2010.

[31] http://www.nvidia.com/page/geforce_6600.html. Information geforce 6600.

[32] Daniel Cederman and Philippas Tsigas. A practical quicksort algorithm for graphics processors. In Dan Halperin and Kurt Mehlhorn, editors, *Algorithms - ESA 2008*, volume 5193 of *Lecture Notes in Computer Science*, pages 246–258. Springer Berlin / Heidelberg, 2008.

[33] Benjamin Bustos, Oliver Deussen, Stefan Hiller, and Daniel Keim. A graphics hardware accelerated algorithm for nearest neighbor search. In *Computational Science – ICCS 2006, volume 3994 of LNCS*, pages 196–199. Springer, 2006.

[34] Osami Yamamoto. An acceleration technique for the computation of voronoi diagrams using graphics hardware. In Osvaldo Gervasi, Marina Gavrilova, Vipin Kumar, Antonio Laganà, Heow Lee, Youngsong Mun, David Taniar, and Chih Tan, editors, *Computational Science and Its Applications – ICCSA 2005*, volume 3480 of *Lecture Notes in Computer Science*, pages 786–795. Springer Berlin / Heidelberg, 2005.

[35] Hanyoung Jang and JungHyun Han. Fast collision detection using the a-buffer. *The Visual Computer*, 24:659–667, 2008.

[36] Satoshi Ohshima, Kenji Kise, Takahiro Katagiri, and Toshitsugu Yuba. Parallel processing of matrix multiplication in a cpu and gpu heterogeneous environment. In *In 7th International Meeting on High Performance Computing for Computational Science (VECPAR'06*, pages 41–50, 2006.

[37] Oliver Kutter, Ramtin Shams, and Nassir Navab. Visualization and gpu-accelerated simulation of medical ultrasound from ct images. *Comput. Methods Prog. Biomed.*, 94:250–266, June 2009.

[38] Bryson R. Payne, Saeid O.Belkasim, G. Scott Owen, Michael C.Weeks, and Ying Zhu. Accelerated 2d image processing on gpus. In Vaidy S. Sunderam, Geert Dick van Albada, Peter M. A. Sloot, and Jack J. Dongarra, editors, *Computational Science – ICCS 2005*, volume 3515 of *Lecture Notes in Computer Science*, pages 256–264. Springer Berlin / Heidelberg, 2005.

[39] Edward Kandrot Jason Sanders. *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison-Wesley Professional, Boston, 2010.

[40] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architexture*. NVIDIA, USA, 2009.

[41] `http://www.intel.com/support/processors/sb/CS-023143.htm#2`. Especificaciones de los procesadores intel, 2010.

[42] `http://theinf2.informatik.uni-jena.de/Lectures/Programming+with+CUDA/WS+2009_2010.html`. Programming with cuda, 2010.

[43] Naga K. Govindaraju, David Knott, Nitin Jain, Ilknur Kabul, Rasmus Tamstorf, Russell Gayle, Ming C. Lin, and Dinesh Manocha. Interactive collision detection between deformable models using chromatic decomposition. *ACM Trans. Graph.*, 24:991–999, July 2005.

[44] Ming C. Lin and Stefan Gottschalk. Collision detection between geometric models: A survey. In *In Proc. of IMA Conference on Mathematics of Surfaces*, pages 37–56, 1998.

[45] P. Jiménez, F. Thomas, and C. Torras. 3d collision detection: A survey. *Computers and Graphics*, 25:269–285, 2000.

[46] Matthew Moore and Jane Wilhelms. Collision detection and response for computer animation. *SIGGRAPH Comput. Graph.*, 22:289–298, June 1988.

[47] Sean Curtis, Rasmus Tamstorf, and Dinesh Manocha. Fast collision detection for deformable models using representative-triangles. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, I3D '08, pages 61–69, New York, NY, USA, 2008. ACM.

[48] C. Ericson. *Real Time Collision Detection*. Morgan Kaufmann, San Francisco, CA, 2005.

[49] Stefan Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina, USA, 2000.

[50] `http://www.udk.com/`. Unreal development kit, 2010.

[51] Xinyu Zhang and Young J. Kim. Interactive collision detection for deformable models using streaming aabbs. *IEEE Transactions on Visualization and Computer Graphics*, 13:318–329, March 2007.

[52] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, New York, NY, USA, 1996. ACM.

[53] Guibas L. J. Mitchell J. S. Barequet G., Chazelle B. and Tal A. Boxtree: A hierarchical representation for surfaces in 3d. In *Computer Graphics Forum*, volume 15, pages 387–396. Blackwell Science Ltd, 1996.

[54] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 47–57. ACM, 1984.

[55] Hanan Samet and Robert E. Webber. Hierarchical data structures and algorithms for computer graphics. part i. *IEEE Comput. Graph. Appl.*, 8:48–68, May 1988.

[56] Michael Paterson and F. Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete and Computational Geometry*, 5:485–503, 1990. 10.1007/BF02187806.

[57] `http://graphics.stanford.edu/software/trimesh/`. Trimesh library, standford university, 2010.

[58] Tomas Möller. A fast triangle-triangle intersection test. *Journal of graphics, gpu, and game tools*, 2(2):25–30, 1997.

[59] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *journal of graphics, gpu, and game tools*, 2(1):21–28, 1997.

[60] Marta Fairén Eduardo Roa, Víctor Theoktisto and Isabel Navazo. Gpu collision detection in conformal geometric space. *Submitted 7/2/2011 to the V Ibero-American Symposium in Computer Graphics, SIACG2011*, 2011.