Construcción de superficies implicitas en 3D a partir de una nube de puntos utilizando Marching Triangles

Diego Arévalo Ovalle*

Octubre 2014

Abstract

Hilton and Illingworth introduced a new surfaced-based approach to triangulation of an implicit surface [1] called Marching Triangles, In this paper we review the algorithm presented by Hilton and propose a modification for the reconstruction of certain surfaces.

Keywords: *Triangulation Delaunay, Marching Triangles, 3D surface.*

Resumen

En este trabajo se presenta una modificación del algoritmo de Marching Triangles presentado por Hilton y Illingworth [1] para la reconstrucción de algunas superficies 3D.

Palabras claves: *Triangulación Delaunay, Marching Triangles, superficies 3D.*

1 Preliminares

En esta sección vamos ver algunos conceptos básicos de forma superficial tanto en geometría como en computación, que nos permitan entender mejor la problemática a la que nos enfrentamos, también vamos a ver como estos conceptos son aplicados.

2 Conceptos Básicos

2.1 Nube de Puntos

En geometría un vértice es un punto donde concurren los dos lados de un ángulo o tres o más planos, la punta de un cono o pirámide o el punto de una curva en la que la curvatura es

*Facultad de Ingeniería y Ciencias básicas. Institución Universitaria Politécnico Grancolombiano, a_ovalle@poli.edu.co

máxima también se conocen como vértice, una nube de puntos está formada por un conjunto de vértices, en este proyecto se trabaja en un espacio tridimensional para representar un vértice en coordenadas cartesianas utilizamos (x, y, z) donde "x" es el punto correspondiente en el eje "X", "y" es el punto correspondiente al eje "Y" y "z" es el punto correspondiente en "Z". Hay diferentes formas de obtener una nube de puntos ya sea utilizando un escáner de superficies que nos permite obtener una cantidad finita de puntos de la superficie escaneada o por medio de métodos computacionales que nos proporcionan una cantidad finita de puntos ya sea de una superficie programada o aleatoriamente delimitando el espacio en donde se encuentran estos puntos.

2.2 Polígono Convexo

Un conjunto de puntos en un plano, los cuales determinan a un polígono se denominan convexo si y solo si al unir dos puntos del conjunto de puntos este está contenido en su totalidad dentro del polígono.

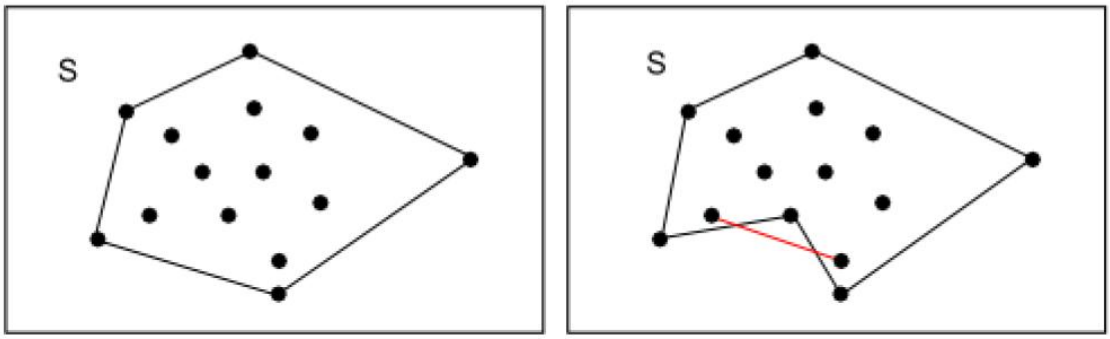

Figure 1: En la imagen se puede observar que el polígono que se encuentra a la izquierda si es convexo porque cumple con la condición, la imagen de la derecha no cumple la condición[2]

2.3 Figuras Geométricas

Para el desarrollo de este proyecto se utilizan diferentes figuras geométricas como los triángulos que se utilizan para la construcción de una malla que a su vez representa a una superficie, los triángulos tienen diferentes propiedades que aplican en forma teoría y práctica. A continuación se describen algunos de los conceptos necesarios para un entendimiento más claro y preciso.

2.4 Orientación

Sea T un triángulo definido por los vértices A,B,C, al unir estos vértices se generan las aristas a,b,c es importante determinar algún tipo de orientación para el mismo, se pueden definir dos tipos de orientación en sentido horario y en sentido anti-horario, en el desarrollo de este

proyecto trabajaremos en sentido anti-horario ya que dependiendo de la orientación se debe tener diferentes criterios de evaluación, en la siguiente figura se puede observar los tipos de orientación.

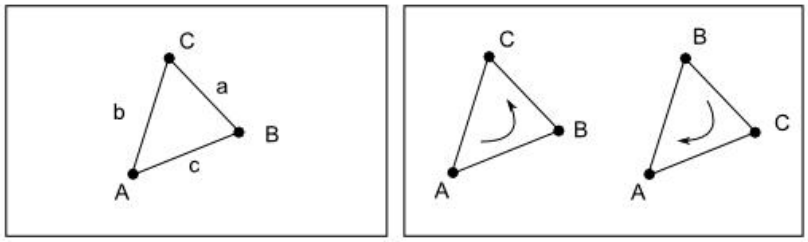

Figure 2: Tipos de orientación de un triangulo[2]

2.5 Circulo Circunscrito

Dado un triángulo T cualquiera, se puede hallar el circulo que pasa exactamente por los tres vértices de T, a este círculo se denomina "Circulo Circunscrito", en la siguiente figura se puede observar con más detalle.

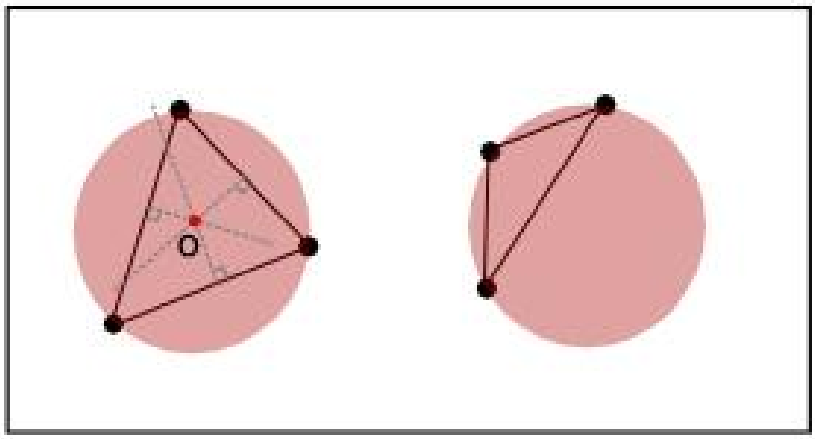

Figure 3: Circulo Circunscrito[2]

2.6 Ubicación de un punto respecto a una recta

Existe un concepto que es muy útil en la triangulación y es el de determinar si un punto está a la derecha o izquierda con respecto a una recta L, esto lo podemos calcular ya que L tiene una orientación en este caso particular L está orientada en sentido anti-horario.

Supongamos que tenemos una recta L, compuesta por dos puntos A,B y supongamos que tenemos un punto C en el mismo plano, deseamos saber si este punto está ubicado a la

derecha o izquierda de L, para lograr esto debemos hallar el valor de la determinante entre A,B,C si el valor de la determinante es mayor a 0 significa que C se encuentra a la derecha de L en caso contrario se encontraría a la izquierda en la figura se puede observar con más detalle.

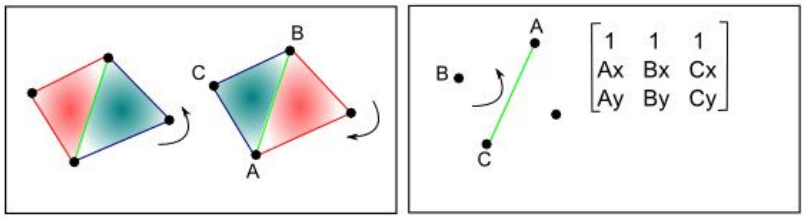

Figure 4: Ubicación de un punto respecto a una recta.[2]

2.7 Ubicación de un punto respecto a un Plano geométrico

Aunque el concepto anterior es importante no es de utilidad en tres dimensiones por esta razón utilizamos otro concepto que se basa en determinar si un punto se encuentra a la derecha o izquierda de un plano geométrico, un plano está definido por dos vectores $\vec{u} = a_x, a_y, a_z$,$\vec{v} = b_x, b_y, b_z$ y un punto $P = c_x, c_y, c_z$, para poder determinar si un punto se encuentra ubicado en la parte derecha o izquierda del plano, se debe utilizar los conceptos de vector normal. Supongamos que tenemos que tenemos un Triángulo $T_{new} = (x_0, x_1, x_2)$ donde x_0, x_1, x_2 son vértices y estamos evaluando la arista $e_0 = x_0, x_1$ y tenemos un punto $p0 = (a_0, a_1, a_2)$ donde a_0 es la coordenada en el eje X, a_1 es la coordenada en el eje Y y a_2 es la coordenada en el eje Z del punto, ahora tenemos que hallar el vector $\vec{k}$ que corresponde a la dirección del plano.

1. Calcular $\vec{u}$
   $\vec{u} = (x_1 - x_0)$
2. Calcular $\vec{v}$
   $\vec{v} = (x_2 - x_0)$
3. Calcular $\vec{x0}$
   $\vec{x0} = (x_0 - t_0)$ donde $t_0 = (0, 0, 0)$
4. Calcular $\vec{k}$
   $\vec{k} = \vec{u} \times (\vec{u} \times \vec{v}) + \vec{u} \times \vec{x_0}$
5. Evaluar punto
   Si $[\vec{k} \cdot (p - x_0)][\vec{k} \cdot (x_2 - x_0)] < 0$ significa que p está en la dirección de $\vec{k}$, en caso contrario se encuentra en la parte opuesta del plano.

3 Computación Gráfica

Computación gráfica es una rama de la informática donde el computador es el encargado de crear o modificar las imágenes de una forma artificial o sintética, los gráficos por computador se conocen desde hace bastantes años pero en la actualidad es muy utilizada en diferentes áreas (Juegos de Computadora, Películas animadas, Medicina, etc.) y una de sus principales ventajas es que todo se realiza en un entorno virtual aunque existen limitaciones como en cualquier otra área en algunos aspectos es más flexible y económica.

4 Triángulo

Un triángulo es un polígono acotado por tres rectas que se cortan entre sí, por lo tanto un triángulo tiene 3 vértices, 3 lados y 3 ángulos interiores, los triángulos se pueden clasificar por la longitud de sus lados (Equilátero, Isósceles, Escaleno) o la amplitud de sus ángulos interiores (Rectángulo, Obtusángulo, Acutángulo), los ángulos de un triángulo siempre suman 180 Grados, los triángulos han sido usados y estudiados a lo largo de la historia humana gracias a esto hoy en día se conocen muchas propiedades y teoremas.

5 Triangulación

La triangulación es un método geométrico utilizado en diferentes áreas y permite unir tres vértices distintos obteniendo un triángulo como resultado, es un método muy útil para resolver diferentes problemas geométricos y todo gracias a las propiedades geométricas, los triángulos son fáciles de construir y realizar diferentes cálculos con ellos por ejemplo para hallar el área que encierra un triángulo se utiliza la siguiente ecuación: $A\triangle = (b * h)/2$, donde "b" es la base y "h" la altura del triángulo.

6 Triangulación de Superficies definidas Implicitamente

Una triangulación es una subdivisión de una área en triángulos [4] y una superficie implícita se define matemáticamente como un conjunto de puntos en el espacio "X" que satisface la ecuación $f(x) = 0$, para visualizar las superficies implícitas se debe encontrar los ceros de la función f y puede ser realizado por poligonización (Polígonos , Pentágonos, etc.) o contacto directo a través del trazado de rayos.

7 Triangulación de Delaunay

Fue inventada en 1934 por el matemático ruso Boris Nikolaevich Delone, este método permite maximizar el ángulo interior de la red de triángulos que unidos delimitan la superficie de un

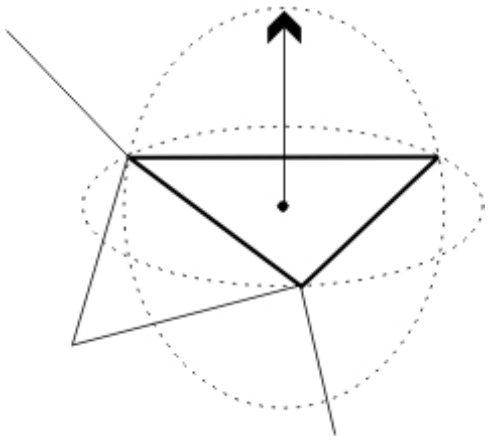

Figure 5: Representación de la condición de Delaunay en 2D. [2]

objeto, esto permite obtener una representación más real del objeto, existen diferentes técnicas para realizar triangulaciones. Para que una triangulación sea considerada de Delaunay, los triángulos que comprenden la triangulación deben cumplir la siguiente restricción.[5]

1. La circunferencia circunscrita al triángulo no debe contener ningún otro vértice que no sea del triángulo.

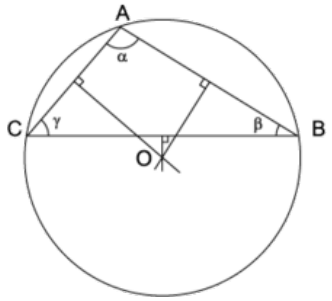

Figure 6: Representación de la condición de Delaunay en 2D.

La triangulación de Delaunay inicialmente fue aplicada en un espacio bidimensional pero se puede aplicar a un espacio tridimensional teniendo en cuenta que ya no se evalúa solo la circunferencia circunscrita al triángulo sino la esfera circunscrita al triángulo que se forma.[3]

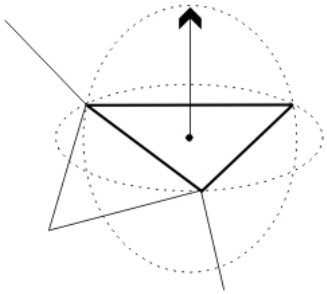

Figure 7: Representación de la condición de Delaunay 3D.

8 Marching Triangles

Marching triangles es una adaptación de la "Triangulación de Delaunay",es una técnica de reconstrucción de superficies geométricas en dos dimensiones que tiene como base una nube puntos, esta nube de puntos puede ser definida por un escáner de objetos que funciona de manera similar a un escáner convencional pero no entrega un duplicado convencional sino una nube puntos del objeto escaneado, también es posible obtener la nube de puntos de una superficie definida implícitamente, existen diferentes técnicas para la reconstrucción de superficies pero en este documento nos centraremos en "Marching Triangles" por las diferentes ventajas que observaremos a lo largo del desarrollo del documento.

Una superficie definida implícitamente es representada matemáticamente como $f(x) = 0$, donde $f(x)$ es el rango de la función y x es un punto en el espacio representado en R^3.

Basados en la restricción de Delaunay para la elaboración de triángulos para la reconstrucción de objetos en tres dimensiones podemos implementar el algoritmo de Marching Triangles, este algoritmo se despliega en una superficie implícita representada por $[f(x), b(x)]$ esta superficie debe ser convexa, este algoritmo procede de la siguiente forma:

9 Descripción del Algoritmo

El algoritmo de "Marching Triangles", es incremental, por lo tanto necesita una base donde comenzar y termina cuando la superficie sea reconstruida, el punto de partida es un triángulo que debe estar sobre la nube de puntos $M(X) = M_0(X_0)$, el algoritmo se detiene una vez se han recorrido todas las aristas. Cada vez que encontrados nuevos puntos y vértices se agregan a la lista. Este modelo $M(X)$ es representado por una lista de vértices y aristas, este algoritmo es implementado en un solo recorrido de nuestra lista de aristas, a medida que se recorren las aristas y se realizan los cálculos para encontrar nuevos triángulos que estén sobre la superficie y cumpla la condición de la Delaunay estas son agregadas a nuestra lista inicial, el algoritmo no termina antes de recorrer y evaluar todas las aristas contenidas en nuestra lista. El algoritmo avanza por evaluar cada arista $e_{bound} = e(x_i, x_j)$ en el límite del

modelo actual M.

9.1 Estimar posición del nuevo vértice

El nuevo vértice se debe proyectar perpendicularmente a la arista y esta arista debe estar en la frontera de nuestro triangulo e_{bound}, la distancia a proyectar es una constante definida por x_{proj}, en el plano modelo del elemento de frontera, $T_{bound} = T(x_i, x_j, x_k)$.

9.2 Evaluar la posición del nuevo vértice

Una vez encontrado el nuevo vértice x_{new} se debe encontrar el punto más cercano de la superficie para todo $x_{proj} : x_{new} = x_{nearest}$ donde $f(x_{nearest}) = 0$.

9.3 Dibujar la malla por la arista de frontera si cumple:

(a) El punto más cercano de la frontera es $b(x_{new}) =' true'$

(b) El producto punto entre la arista de frontera y nuevo vértice sea menor a 0, $T_{bound} : n_{bound}.n_{new} < 0$.

9.4 Evaluar con Delaunay para superficies 3D

Aplicar la condición de Delaunay al nuevo triangulo generado T_{new}.

9.5 Cumple condición de Delaunay para superficies 3D

Se construye el nuevo triangulo T_{new}.

(a) Agregar nuevo vertice x_{new} a la lista de vértices.

(b) Agregar nuevo triangulo T_{new} a la malla M.

(c) Agregar nuevas aristas $e(x_j, x_{new})$ y $e(x_{new}, x_i)$ al final de la lista de aristas.

9.6 No cumple condición Delaunay para superficies 3D

Entonces aplicar pasos 4 y 5 para los vértices de frontera adyacentes, $T_{new} = T_{prev} = T(x_i, x_j, x_{prev})$ ó $T_{new} = T_{next} = T(x_i, x_j, x_{next})$, estos pueden ser el nuevo o el siguiente.

9.7 Si T_{new}, T_{prev} y T_{next} fallan

Si T_{new}, T_{prev} y T_{next} fallan la condición local de Delaunay para superficies 3D, si la esfera circunscrita para $T_{new}(\vec{x}_i, \vec{x}_j, \vec{x}_{new})$ contiene el límite del triángulo, $T_{overlap}$, sobre una parte existente del modelo, $M(X)$ con la misma orientación del límite del triángulo, T_{bound}, tal que $n_{bound}.n_{overloap} > 0$, entonces aplicar pasos 4 y 5 con $T_{new} = T(\vec{i}, \vec{j}, \vec{overloap})$ donde $\vec{overload}$ es el vertice más cercano al borde en $T_{overloap}$.

9.8 Evaluar el vértice e_{bound}

Evaluamos el vértice e_{bound} termina cuando no se pueden agregar más triángulos al modelo, M.

10 Modificación de Marching Triangles

El enfoque original de este proyecto era la implementación del algoritmo Marching Triangles pero en el desarrollo del mismo se fue adaptando según los avances en el dominio del tema que se iba teniendo, entre las modificaciones se puede observar que el algoritmo original al estimar la posición del nuevo vértice lo hace sobre el vector normal del arista evaluada, en este caso en particular se calcula teniendo en cuenta el vértice opuesto del triángulo que contiene la arista.

Para realizar la construcción a partir de una nube de puntos se sigue el siguiente proceso teniendo en cuenta que el proceso de reconstrucción termina una vez se recorran todas las aristas generadas:

10.1 Estimar nuevo vértice x_{new}

El nuevo vértice se proyecta sobre la línea que sale desde el vértice opuesto a la arista e_{bound} y pasa por el punto medio de la arista e_{bound}, teniendo en cuenta que la arista debe ser de frontera y el vértice opuesto se obtiene del triángulo T_{new} que contiene a e_{bound}, la distancia de proyección es una constante definida por x_{proj}.

10.2 Estimar punto más cercano de la nube de puntos al nuevo vértice

Para estimar el punto más cercano hacemos un recorrido por nuestra lista de puntos y excluimos los vértices de T_{new} y de e_{bound}, después calculamos la distancia $d_{temp} = \sqrt{x_{new} - x_{temp}{}^2}$ entre x_{new} y x_{temp} donde x_{temp} es el punto que estamos evaluando en ese momento, para que x_{temp} sea tenido en cuenta la distancia entre x_{temp} y x_{new} debe ser la menor posible y x_{temp} debe estar en la misma parte del plano que x_{new}.

10.3 Evaluar condición de Delaunay 3D para T_{new}

Aplicar la condición de Delaunay 3D a T_{new}.

10.4 Si T_{new} cumple Delaunay 3D

Se dibuja T_{new}.

a. Agregar nuevo vértice x_{new} a la lista de vértices.

b. Agregar nuevo triangulo T_{new} a la malla.

c. Agregar nueva arista $e(x_j, x_{new})$ si no está en la lista de aristas en caso contrario se actualiza la arista como arista de centro.

d. Agregar nueva arista $e(x_{new}, x_i)$. si no está en la lista de aristas en caso contrario se actualiza la arista como arista de centro.

10.5 Si T_{new} no cumple Delaunay 3D

Aplicar pasos 2.3.1 y 2.3.2 a los vértices de frontera adyacentes de $T_{new} = T_{prev} = T(x_i, x_j, x_{prev})$ ó $T_{new} = T_{next} = T(x_i, x_j, x_{next})$, estos son los vértices siguientes y anterior.

10.6 Si T_{new}, T_{next}, T_{prev} fallan

Si T_{new}, T_{next}, T_{prev} fallan la condición de Delaunay para superficies 3D, entonces se debe tener en cuenta el vértice sobrepuesto *overloap* para hacer $T_{new} = T(x_i, x_j, overloap)$, donde *overloap* es el vértice más cercano contenido en la condición de Delaunay.

10.7 Evaluar siguiente arista e_{bound}

Evaluamos la arista e_{bound} termina cuando se puedan generar más aristas al modelo.

11 Análisis y diseño

Se comenzó el análisis teniendo claro que la figura geométrica a utilizar para la reconstrucción de la superficie eran triángulos, con esta importante base se comenzó a profundizar en las diferentes propiedades de los triángulos y sus aplicaciones prácticas.
Para el diseño de la solución se tuvieron en cuenta los diferentes algoritmos de triangulación pero se hizo énfasis en el algoritmo propuesto por A. Hilton y J. Illingworth, "Marching Triangles"[1] por ser una adaptación del algoritmo de "Triangulación de Superficies Implícitas de Delaunay".
Es importante aclarar que el algoritmo que se implemento es una adaptación o modificación

de “Marching Triangles”, la primer modificación que se realizo fue la de estimar el nuevo vértice, el cual no se proyecta perpendicularmente desde el punto medio de la arista de frontera a una distancia definida por una constante de proyección sino que se traza una línea desde el vértice opuesto a la arista y pasa por el punto medio de la arista para ser proyectado una distancia definida por la constante de proyección, con este cambio ya no se tendría que calcular si el nuevo vértice esta sobre el mismo plano que el triángulo.

11.1 Requerimientos

Es necesario tener en cuenta los siguiente factores para obtener el resultado esperado respecto al algoritmo.

1. Definir una constante de proyección, preferiblemente que este entre 0.001 y 0.1
2. El triángulo inicial debe estas ubicado sobre la nube de puntos y cumplir la condicion de Dalaunay.
3. Si se trabaja con una nube de puntos a una escala muy pequeña se debe multiplicar cada punto por un factor para agrandar la superficie a reconstruir y visualizar mejor la superficie.
4. La nube de puntos inicial se debe cargar desde una archivo ”OBJ´´ se debe especificar el nombre del archivo junto con la extensión del mismo ejemplo ”miarchivo.obj´´, la librería solo carga los vértices del archivo.

11.2 Diagrama de clases

Para la codificación del algoritmo era necesario definir las estructuras de datos apropiadas, por agilidad y practicidad se utilizaron las siguientes clases para representar los datos necesarios del problema y para almacenar las colecciones de objetos se utilizaron ArrayList.

11.3 Descripción general diagrama de clases

Las clases (Point3D, Edge3D, Triangle3D) cuentan con sus modificadores de acceso (Setter and Getters).

1. Point3D
   Esta clase se encarga de representar un vértice definido por (x,y,z) y uno de sus métodos relevantes es evaluar si dos vértices son iguales que se llama isEquals(Point3D p) y retorna un booleano.
2. Edge3D
   Esta clase contiene la información de una arista y es vital para el desarrollo del algoritmo porque a través de sus métodos se puede obtener el punto medio que se utiliza

para calcular el nuevo vértice estimado, también se debe destacar los métodos que permiten obtener los triángulos que colindan con la arista por derecha o izquierda.

3. Triangle3D
   La clase Triangle3D aparte de tener la información relevante de un triangulo en R^3, nos permite evaluar la condicio de Delaunay para un triangulo en particular, obtener una arista o vertice, como obtener el punto opuesto a una arista con su metodo getPointProyectar(Edge3D e).

4. Main
   En este caso en particular esta clase se encarga de la lógica del negocio, es la clase que realiza la ejecución de los métodos en un orden específico para realizar la reconstrucción de la superficie.

12 Complejidad del Algoritmo

La complejidad computacional del algoritmo de reconstrucción de la superficies definidas implícitamente puede ser definida en términos del número de funciones de evaluación. El tiempo de la complejidad depende del número de aristas, N_e que contenga la solución, la cantidad de vértices N_n, la cantidad de triángulos N_f y la cantidad de identificadores N_h.

$$N_h : N_n - N_e + N_f = 2 - 2N_h$$

Por ende si el numero de identificadores es pequeño el numero de triangulos y vertices es del mismo orden entonces el numero de aristas es $O(N_n) = O(N_e) = O(N_f)$. Un algoritmo convencional de triangulación de fuerza bruta tiene una complejidad del orden de $O(n^3)$, al ser este un algoritmo incremental de inserción de puntos que tiene como particularidad que mantiene una solución parcial del problema desde el comienzo y garantiza que cada nueva triangulación que se agregue cumple con la condición de Delaunay, obtenemos una complejidad del orden de $O(n^2)$ aunque la complejidad de una triangulación de Delaunay es del orden de $O(n(log(n)))$ este algoritmo necesitaría algunos ajustes en cuanto a las estructuras de datos para poder mejorar la complejidad del mismo por falta de tiempo este punto se tendrá encuentra en un trabajo futuro.

Table 1: Comparación de Superficies Reconstruidas

Objeto	Puntos	Aristas	Triangulos	Iteraciones	Depth	Proy	Factor	Tiempo
Toro	225	940	743	939	241	0.01	40	10seg
Esfera	362	1082	721	1081	246	0.01	300	10seg
Dragon	50000	209247	169369	209246	56483	0.01	300	1hora y 15min

13 Resultados Obtenidos

13.1 Reconstrucción de una esfera

El desarrollo del algoritmo fue planteado para la reconstrucción de una esfera por las propiedades geométricas teniendo en cuenta que una vez terminada la librería se podría utilizar para la reconstrucción de cualquier otra superficie.
Inicialmente no se tomó el triángulo base sobre la superficie de la esfera sino que cada uno de los vértices del triángulo base se encontraran sobre la esfera esto género un retraso significativo en el desarrollo del proyecto, aunque se obtuvieron resultados muy interesantes y se recordaron varios conceptos geométricos, para lograr esta imagen no se utilizo una nube de puntos sino se calcula el nuevo vértice mas cercano a la superficie de la esfera [Figura 5.1].

Después de analizar detenidamente el algoritmo de "Marching Triangles" [1], se concluyó que el triángulo base debe estar sobre la superficie, porque a partir de este triángulo hay una expansión por cada una de las aristas hasta lograr la reconstrucción de la misma, en la siguiente imagen se puede observar el proceso generando 35 aristas [Figura 5.2].

13.2 Reconstrucción de Dona o Toro

Con el objetivo de tener un punto de comparación y poder verificar el comportamiento del algormitmo al reconstruir una superficie diferente a una se esfera se decidio reconstruir un Toro o Dona, en las primeras Decidi reconstruir esta superficie para verificar el comportamiento del algoritmo, en un principio se encontraron un par de inconsistencias, hay que tener en cuenta que la nube de puntos se obtuvo de fuentes en internet, y por esta razon los puntos originales son multiplicados por un escalar para poder ampliar la superficie.

13.3 Reconstrucción de superficie de un Dragon

Para la reconstrucción del dragon la nube de puntos fue de 50.000, los puntos tenian una escala pequeña y fueron multiplicados por un escalar para aumentar el area del dragon, la reconstrucción de esta superficie toma bastante tiempo aproximadamente una hora como minimo en una maquina que cuente con 16GB en Ram y 8 Cores de procesamiento.

References

[1] Marching Triangles,A. Hilton and J. Illingworth,*Marching Triangles: Delaunay Implicit Surface Triangulation*,1996

[2] Torres - Cristian Luis y Wilgenhoff - Cristian Pedro,*Generador de Mallas Triangulares*,2009

[3] Pierre-Alain Fayolle,*Polygonization of implicit surfaces using Delaunay*, technical report, 2009

[4] Marc Fournier,*Surface Reconstruction: An Improved Marching Triangle Algorithm for Scalar and Vector Implicit Field Representations*,2009

[5] M. Fisher - B. Springborn - P. Schröder and A. I. Bobenko,*An Algorithm for the Construction of Intrinsic Delaunay Triangulations with Applications to Digital Geometry Processing*,2007

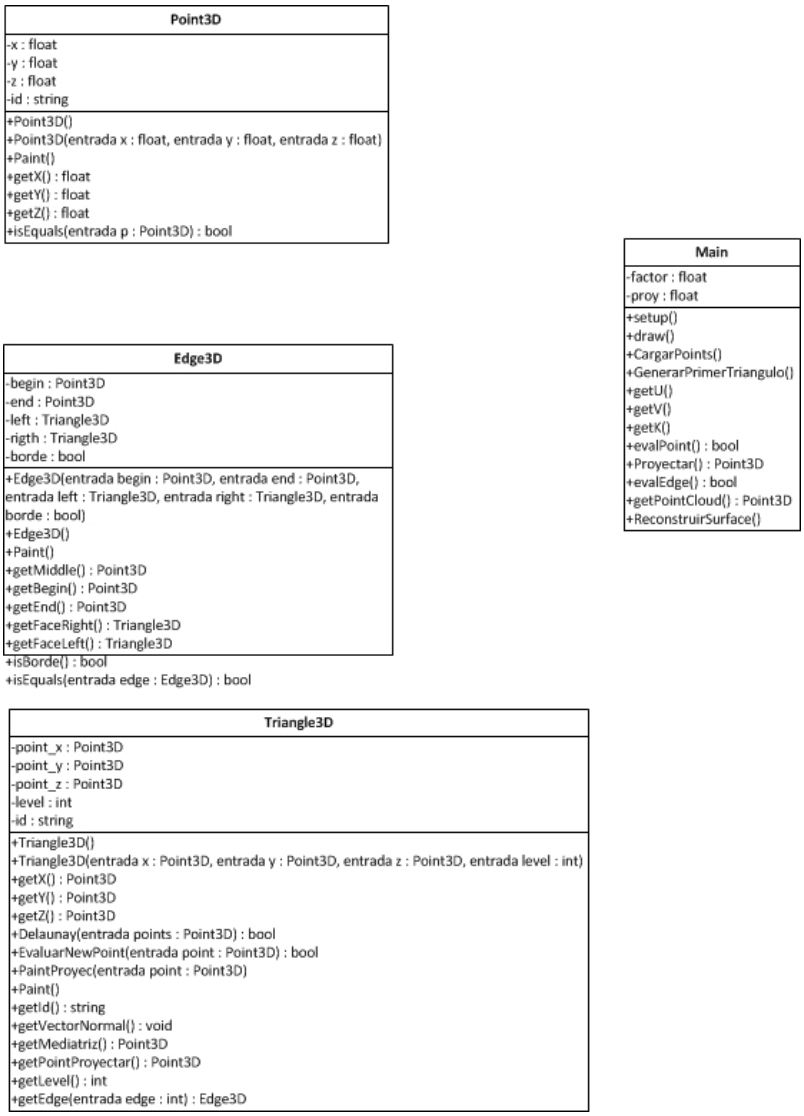

Figure 8: Diagrama de clases

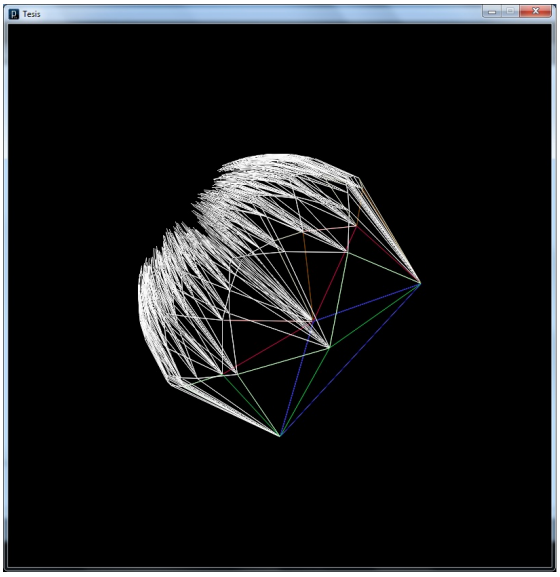

Figure 9: Reconstrucción con una profundidad de triángulos de 5.

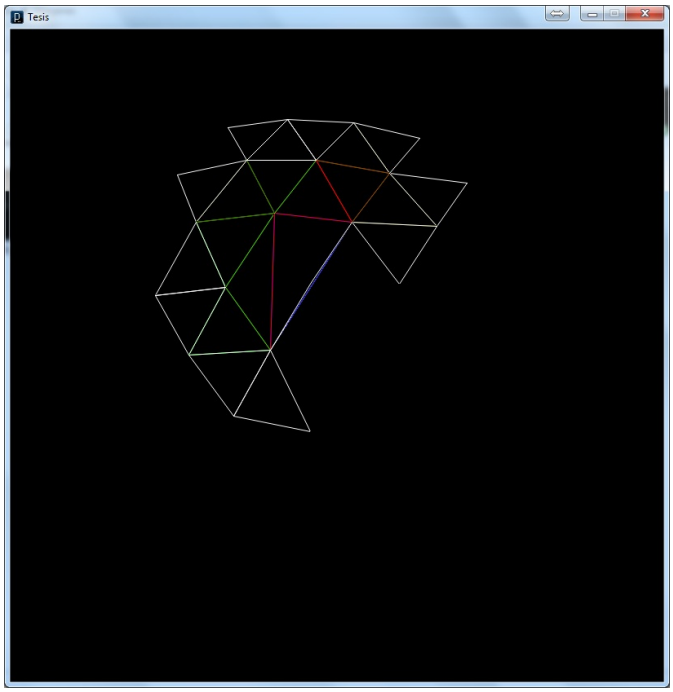

Figure 10: Esfera con 35 aristas evaluadas

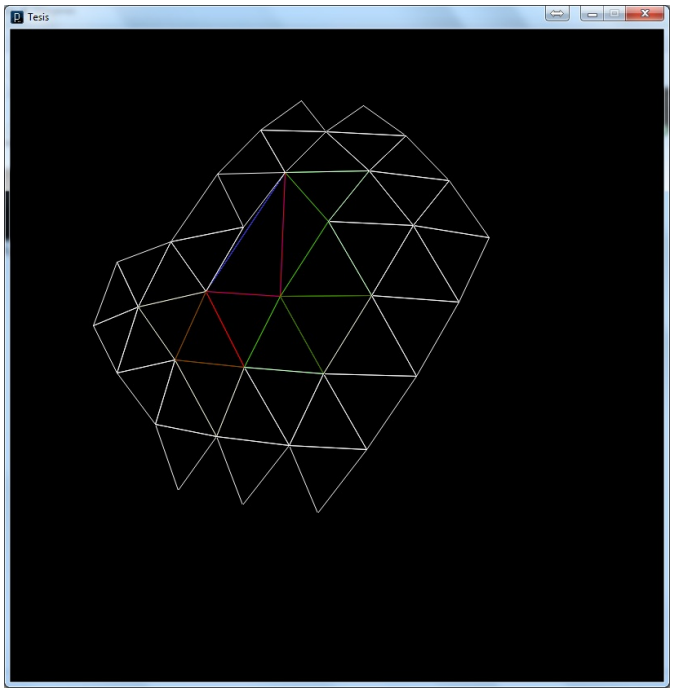

Figure 11: Esfera con 50 aristas evaluadas

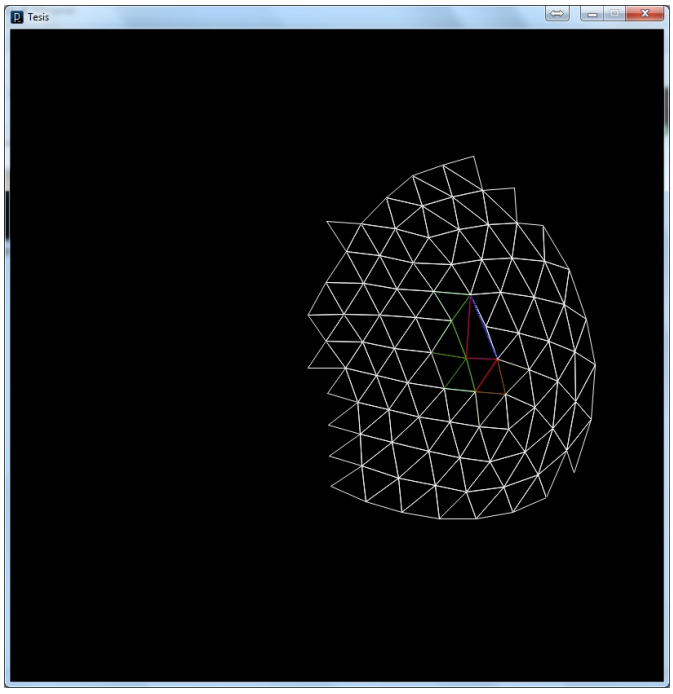

Figure 12: Esfera con 100 aristas evaluadas

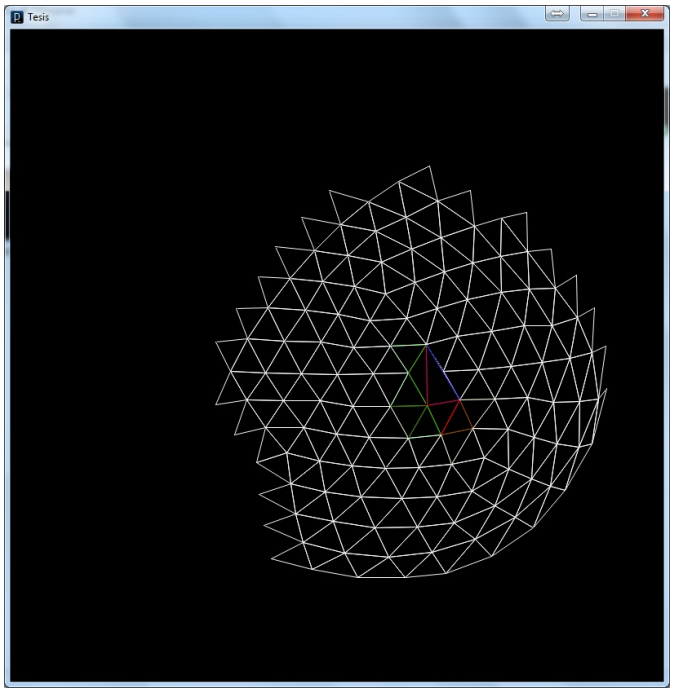

Figure 13: Esfera con 200 aristas evaluadas

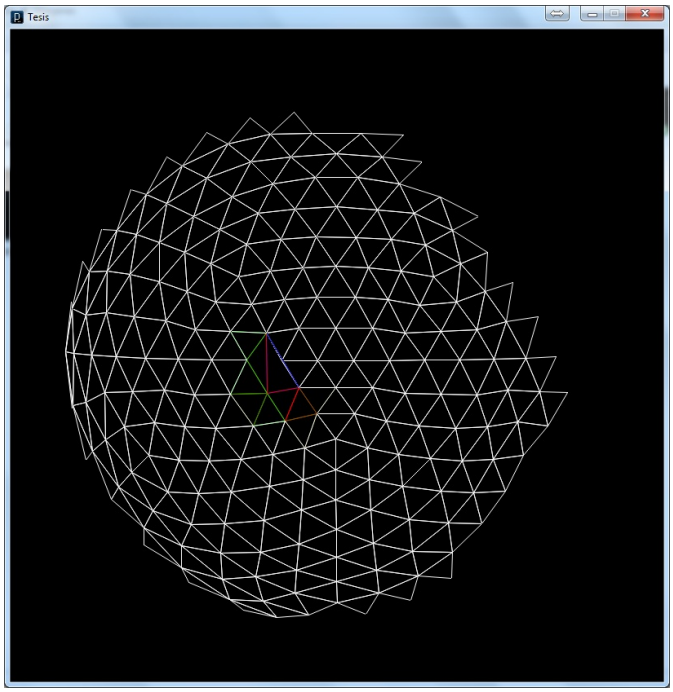

Figure 14: Esfera con 500 aristas evaluadas

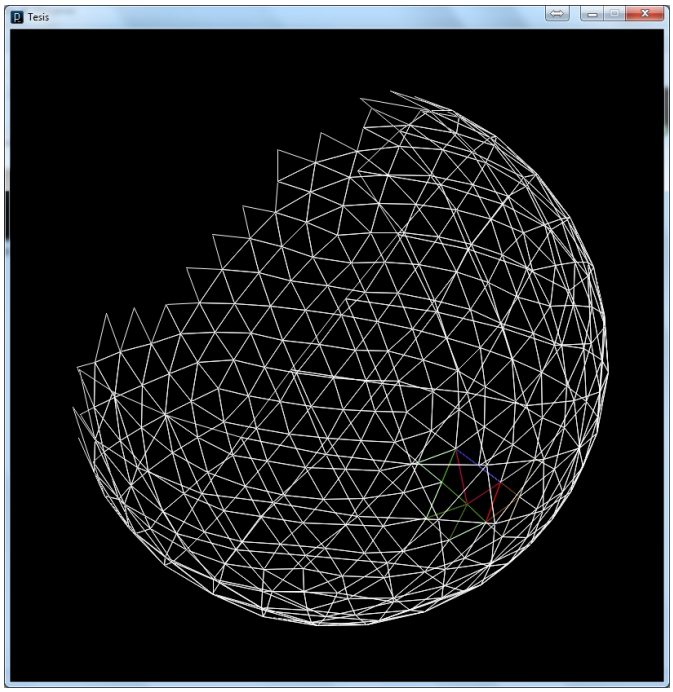

Figure 15: Esfera con 950 aristas evaluadas

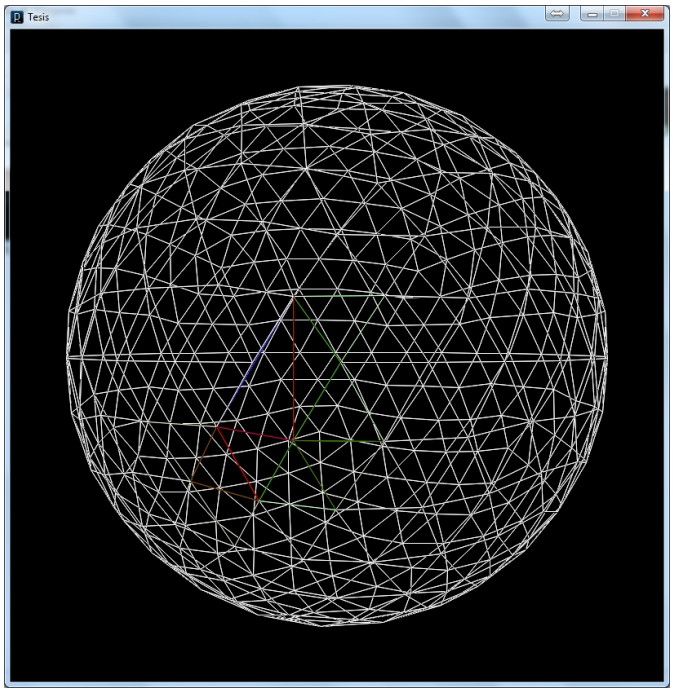

Figure 16: Esfera con todas las aristas evaluadas

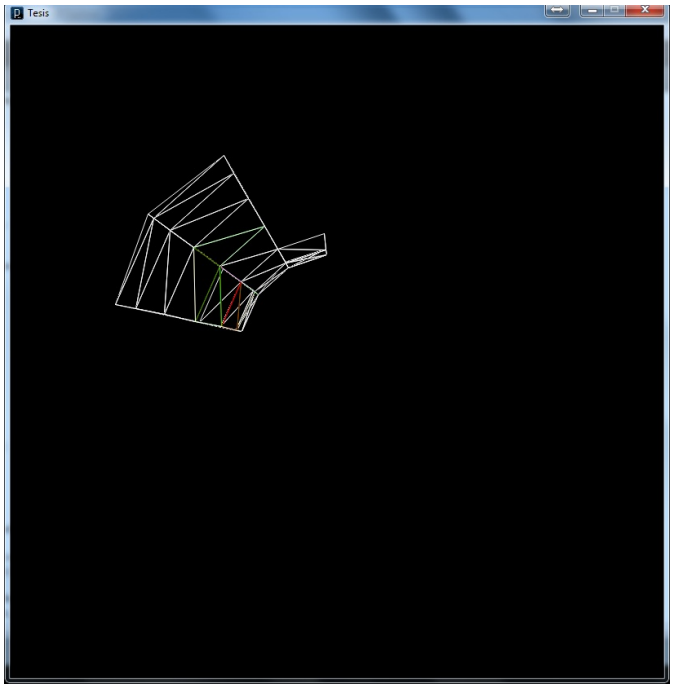

Figure 17: Reconstrucción con 15 arista evaluadas

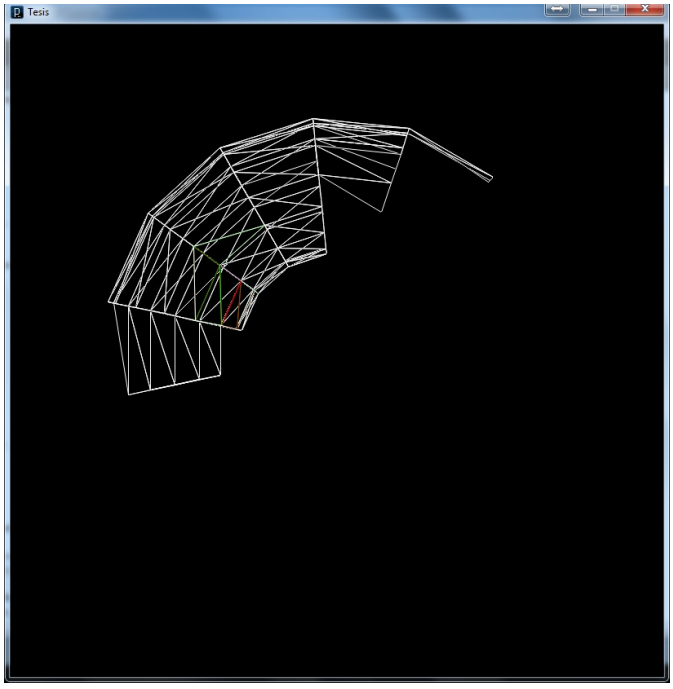

Figure 18: Reconstrucción con 200 arista evaluadas

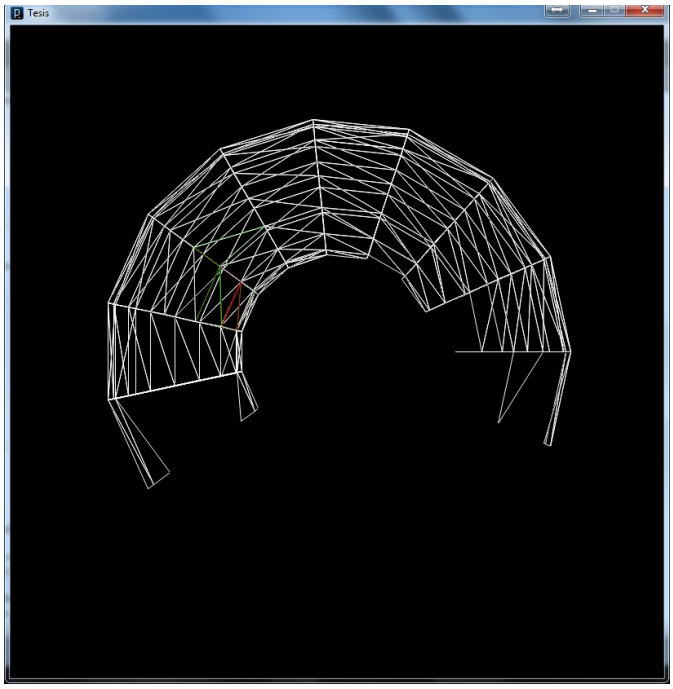

Figure 19: Reconstrucción con 450 arista evaluadas

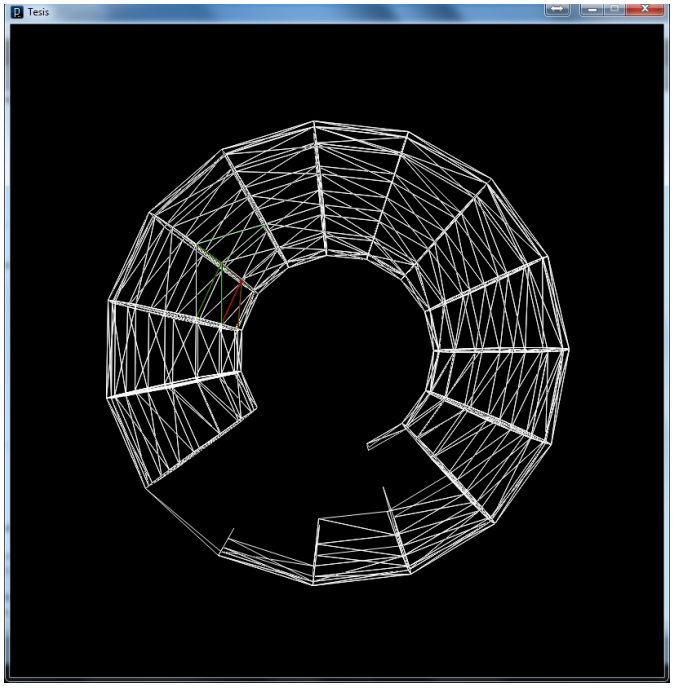

Figure 20: Reconstrucción con 850 arista evaluadas

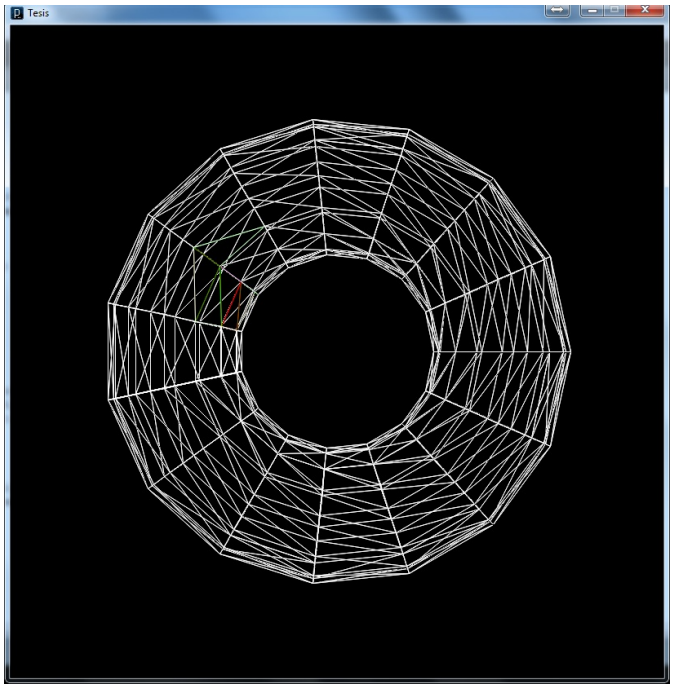

Figure 21: Reconstrucción total

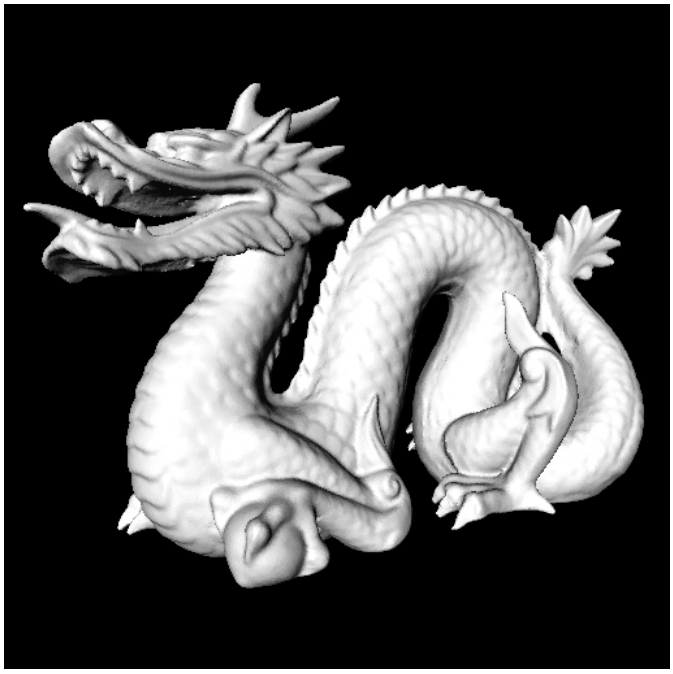

Figure 22: Imagen Original

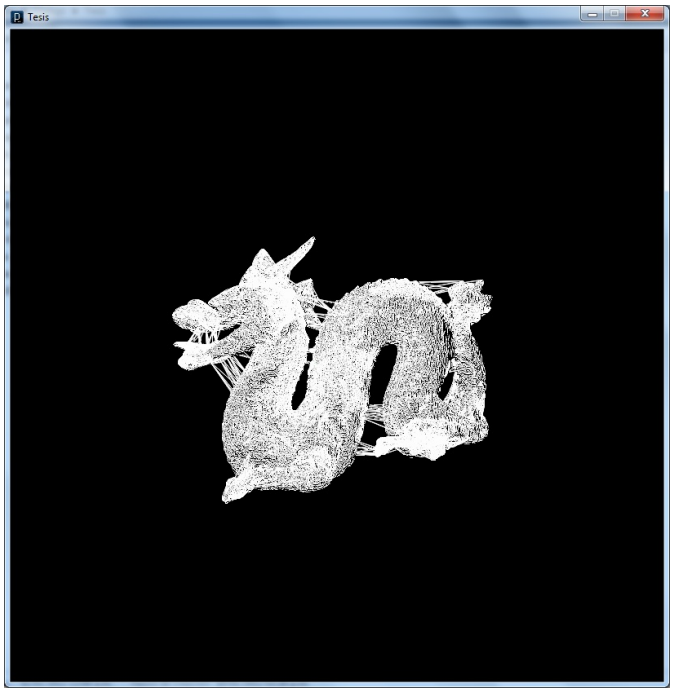

Figure 23: Factor de multiplicación por 100

Figure 24: Factor de multiplicación por 300

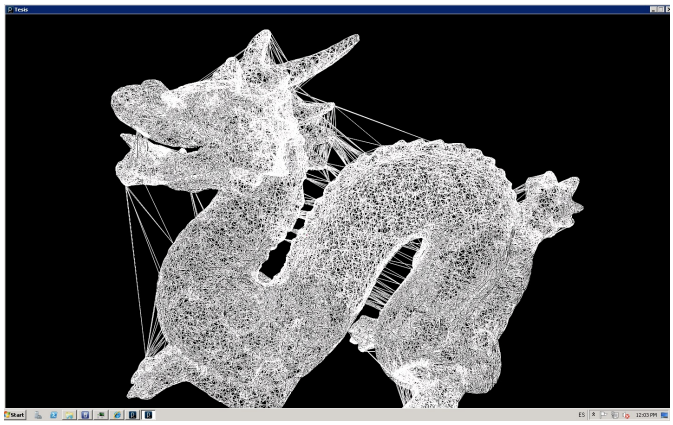

Figure 25: Factor de multiplicación por 300 parte de atras