UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

MARCO DE EXPERIMENTACIÓN PARA ALGORITMOS DE REFINAMIENTO DE TRIANGULACIONES EN 2D

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

ÁLVARO MARTÍN FAÚNDEZ REYES

PROFESOR GUÍA:
MARÍA CECILIA RIVARA ZÚÑIGA

MIEMBROS DE LA COMISIÓN:
NANCY VIOLA HITSCHFELD KAHLER
ALEXANDRE BERGEL

SANTIAGO DE CHILE
NOVIEMBRE 2010

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN
POR: ÁLVARO MARTÍN FAÚNDEZ REYES
FECHA: 01/12/2010
PROF. GUÍA: SRA. MARÍA CECILIA RIVARA ZÚÑIGA

MARCO DE EXPERIMENTACIÓN PARA ALGORITMOS DE REFINAMIENTO DE TRIANGULACIONES EN 2D

El uso de elementos finitos para analizar fenómenos físicos modelados por ecuaciones diferenciales parciales requiere de una discretización del dominio, como lo son las triangulaciones en dos dimensiones. En este contexto se distinguen tres problemas: generar una triangulación a partir de un conjunto de vértices y segmentos, refinar una malla y mejorar la calidad de una malla.

Los algoritmos que refinan triangulaciones Delaunay en general se basan en seleccionar nuevos puntos y realizar una inserción Delaunay de éstos. Los criterios usados para comparar algoritmos se basan en la cantidad de inserciones que realizan, la cantidad de triángulos generados y el tiempo de ejecución. Sin embargo, es difícil encontrar implementaciones que realicen las comparaciones bajo un mismo ambiente y condiciones.

En esta memoria se ha diseñado un marco de experimentación que permite investigar y comparar algoritmos de refinamiento Delaunay dentro de un mismo ambiente. Se propone un proceso de refinamiento genérico y se desarrolla una herramienta que, haciendo uso de patrones de diseño de programación con orientación a objetos, implementa el proceso con la flexibilidad de poder extender la herramienta con nuevos algoritmos de forma simple. Se provee una interfaz gráfica que permite seguir el proceso de refinamiento en una forma clara y didáctica.

La herramienta genera resultados comparables con los resultados de Triangle, una herramienta de refinamiento Delaunay rápida y eficiente, pero limitada en su extensibilidad y usabilidad. La extensibilidad de la herramienta se puso a prueba implementando los siguientes criterios de selección de puntos asociados a triángulos de mala calidad: Circuncentro, Off-Center, Lepp - Punto Medio, Lepp - Centroide y Lepp - Bisección (No Delaunay). Se evalúan también técnicas de priorización en el procesamiento de los triángulos de mala calidad.

Se experimentó con un algoritmo nuevo, Lepp -Circuncentro, el cual presentó un buen rendimiento con las mallas estudiadas, alcanzando en algunos casos una exigencia de 37º como ángulo interior mínimo. Se estudiaron criterios de priorización en la selección de triángulos de mala calidad, concluyendo que el comportamiento de los algoritmos de tipo Lepp es independiente del uso de técnicas de priorización. En cambio, el algoritmo Off-Center aumenta considerablemente el número de puntos insertados si no se utiliza un orden de procesamiento adecuado.

AGRADECIMIENTOS

A mis padres, hermana y hermanos, quienes han estado a mi lado en toda decisión que he tomado en este largo proceso y han sido un respaldo en todo momento. Esta memoria es fruto de su incondicional apoyo.

A mis amigos, quienes hicieron que este camino fuera alegre y gratificante.

A mi profesora guía, quien me tuvo fe y paciencia, y siempre estuvo disponible para sacar la memoria adelante.

Y a Javiera, por su ayuda, compañía y ser siempre una luz de esperanza.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

El uso de elementos finitos para analizar fenómenos físicos modelados por ecuaciones diferenciales parciales requiere de una discretización del dominio en el cual se trabaja. En este contexto, el refinamiento de mallas geométricas en dos dimensiones es fundamental. En esta memoria se diseña e implementa una herramienta para estudiar el comportamiento de los algoritmos desarrollados en esta área, conocer los resultados de los algoritmos y establecer en base a estos las diferencias en las técnicas que utilizan.

En el dominio de triangulaciones en dos dimensiones, se distinguen los siguientes escenarios:

(i) Generación de mallas

A partir de una descripción de puntos y segmentos restringidos, generar una triangulación válida que contenga los puntos como vértices y que incluya los segmentos restringidos como aristas. La triangulación resultante puede ser de tipo Delaunay o alguna de sus variaciones.

(ii) Mejoramiento en la calidad de una malla

Eliminar los triángulos que provocan problemas en la discretización, como lo son los triángulos con ángulos pequeños, insertando nuevos puntos con el objetivo de disolver éstos y crear nuevos triángulos con mejores ángulos.

(iii) Refinamiento de una malla

Mejorar la malla desde el punto de vista de una aplicación externa, en sectores donde el estudio del fenómeno lo necesite, eliminando triángulos e insertando nuevos puntos que mantengan o mejoren la calidad de la malla con nuevos triángulos, y así obtener resultados de mayor precisión del fenómeno en estudio. En este ámbito se manejan dos tipos de refinamiento:

- Refinamientos Delaunay, basados en agregar sucesivamente nuevos puntos a la triangulación con inserciones que mantengan la condición Delaunay de la triangulación (Inserciones Delaunay)

- Refinamientos con particiones, basados en realizar sucesivas particiones anidadas de triángulos

Para mejorar la calidad de la malla o al refinar para tener una mayor densidad de triángulos, los algoritmos de refinamiento se distinguen entre sí por las siguientes decisiones que realizan en cada iteración

- Criterio de selección de nuevo punto a insertar a la malla a partir de un triángulo propuesto para refinamiento

- Criterio de inserción del nuevo punto (con particiones o con inserciones Delaunay)

Además, es relevante el orden en que se refinan los triángulos, ya que un orden en particular en el ordenamiento de los triángulos seleccionados para refinar puede traducirse en un mayor o menor número de inserciones.

Junto a esto, deben ser consideradas las restricciones que puede tener la malla:

- Bordes de la malla, los cuales limitan la inserción de nuevos puntos a un dominio restringido

- Aristas restringidas, es decir, segmentos de recta que deben estar presente obligatoriamente en la triangulación final, representados como la arista de un triángulo o como resultado de la unión de múltiples aristas de triángulo

1.1. Motivación

Las herramientas disponibles que usan algoritmos de refinamiento no están diseñadas para ser fácilmente modificadas o para integrar nuevos algoritmos. Esta memoria se basa en las siguientes motivaciones:

- Proveer una herramienta práctica para estudiar los algoritmos de refinamiento y que permita realizar desde pequeñas modificaciones, como decidir qué triángulos refinar, hasta grandes cambios como podría ser estudiar y añadir un nuevo algoritmo por completo.

- Proveer una herramienta gráfica que permita visualizar en todo momento lo que ocurre con los algoritmos, de modo que pueda seguirse el refinamiento paso a paso, identificando ocurrencias y casos especiales que puedan ser de gran interés para el estudio y preparación de algoritmos mejorados.

- Utilizar los conocimientos de algoritmos y diseño adquiridos en el transcurso del pregrado, los cuáles fueron adquiridos siempre en ambientes de prueba y con fines pedagógicos.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo de esta memoria es entregar un marco de experimentación de algoritmos que considere la implementación de diferentes algoritmos de refinamiento de triangulación en dos dimensiones y las técnicas que estos algoritmos utilizan, y así tener un ambiente que permita comparar la eficacia y rendimiento de estos algoritmos frente a distintas entradas y bajo las mismas condiciones, y que además permita combinar las diferentes técnicas para formular nuevos algoritmos e identificar las mejores prácticas de refinamiento.

1.2.2. Objetivos Específicos

Del objetivo general se desprenden los siguientes objetivos específicos:

1. Diseñar el marco de experimentación que permita comparar, implementar y combinar diferentes criterios y técnicas de refinamiento.

2. Diseñar las estructuras de datos que se usarán para dar soporte a la topología de las triangulaciones y a la relaciones de vecindad entre los elementos topológicos (vértices, aristas y caras)

3. Implementar los siguientes algoritmos de refinamientos de triangulaciones en dos dimensiones:

    - Circuncentro - Delaunay

- Lepp - Centroide - Delaunay
- Lepp - Punto Medio - Delaunay
- Off-Center - Delaunay
- Lepp - Bisección

4. Implementar el marco de experimentación que permita realizar estudios sobre los algoritmos implementados y realizar modificaciones en las decisiones que éstos toman para refinar una triangulación

5. Estudiar distintos criterios en la inserción de nuevos puntos en la triangulación, especialmente cuando la inserción se lleva a cabo en el borde de la malla.

1.3. Estructura de la Memoria

El resto de la memoria se organiza como sigue:

El capítulo 2 presenta una revisión bibliográfica que introduce definiciones y conceptos claves que fueron usados y aplicados en el desarrollo del marco de experimentación y son necesarios para comprender los elementos que estarán presentes durante todo el documento.

A continuación, en el capítulo 3 se propone el diseño general del marco de experimentación, que consiste en un proceso de refinamiento genérico estructurado en cinco pasos bien definidos, derivado del estudio e implementación de los algoritmos de la revisión bibliográfica. A partir de la definición de dicho proceso, se establecen las estructuras de datos que se estiman necesarias y suficientes para proponer un diseño basado en patrones de orientación a objetos, que permita una

extensibilidad elegante y simple del sistema al momento de querer ingresar nuevos algoritmos. Junto a esto, se presenta la aplicación desarrollada.

En el capítulo 4 de experimentación comienza con una validación del proceso de refinamiento propuesto, comparando sus resultados con los resultados sobre las mismas entradas procesadas por la herramienta Triangle. A continuación se procede a extender el marco de experimentación con nuevas técnicas en algunos de los pasos definidos en el proceso de refinamiento. Específicamente, se agregan dos nuevos métodos de selección de triángulo para refinar, uno basado en elegir primero los triángulos de mala calidad que estén en los bordes de la malla y otro basado en dar prioridad a los triángulos de mala calidad con menor circunradio. Además, se agrega un nuevo método en el paso de selección de un nuevo punto para insertar, denominado "Lepp - Circuncentro - Delaunay", con muy buenos resultados en las mallas de ejemplo que se utilizaron para refinar.

En el capítulo 5 se incluyen las conclusiones de esta memoria, analizando los objetivos propuestos y los resultados obtenidos.

En el capítulo 6 se analiza las posibles mejoras que podría tener la herramienta en un trabajo futuro.

CAPÍTULO 2

REVISIÓN BIBLIOGRÁFICA

Las diversas aplicaciones de la computación gráfica, tales como aplicaciones CAD, de modelamiento de sólidos y de simulaciones numéricas, requieren una discretización de objetos complicados en pequeñas elementos básicos que permitan realizar procesos sobre dichos elementos, los cuales usualmente son representados mediante elementos geométricos dentro de un dominio acotado. En esta memoria, los procesos estudiados son algoritmos de refinamiento de mallas, con triángulos como elementos geométricos y acotados al dominio de un plano en dos dimensiones, los cuáles son esenciales en la simulación de fenómenos modelados con ecuaciones de diferencias parciales y con resultados aproximados con métodos de elementos finitos.

2.1. Triangulación 2D

Se define **vértice** como un par ordenado (x, y) con $x, y \in \mathbb{R}$

Se define **triángulo** como una 3-tupla de vértices no colineales. Los segmentos de recta que unen los vértices son denominados **aristas**.

Se define **triangulación**[1] **válida** como un conjunto de triángulos que cumple las siguientes condiciones:

1 Para efectos de esta memoria, triangulación o malla se referirán indistintamente a una triangulación en dos dimensiones.

(i) La intersección del interior de dos triángulos distintos siempre será vacía

(ii) La intersección de dos triángulos vecinos puede una y sólo una de las dos opciones siguientes:

a. un vértice

b. dos vértices y el segmento de recta que une ambos vértices

(iii) La unión de todos los triángulos forma un conjunto conexo

En una triangulación, se define como **arista restringida** a los segmentos de recta que deben estar presentes obligatoriamente en la triangulación. Los **bordes de malla** son un caso particular de aristas restringidas con la particularidad que sólo pueden estar presentes en triángulo.

2.2. Calidad de una Triangulación

Una características de las mallas utilizadas en elementos finitos, es que la densidad de triángulos en distintos sectores de la malla varía de acuerdo a la necesidad de precisión en los resultados que se requiera para dichos sectores. Generar una malla con más triángulos o con más vértices que la que se tiene inicialmente no es un proceso complicado. Un ejemplo trivial consistiría en elegir cualquier triángulo en un sector en que se requiera mayor densidad de la malla y reemplazarlo con los tres triángulos que forma la unión del incentro con los vértices del triángulo (Figura 1). Sin embargo, la malla resultante de un algoritmo como el anterior puede producir resultados desastrosos en la aplicación que se utilice, debido a la dispar forma de los triángulos generados.

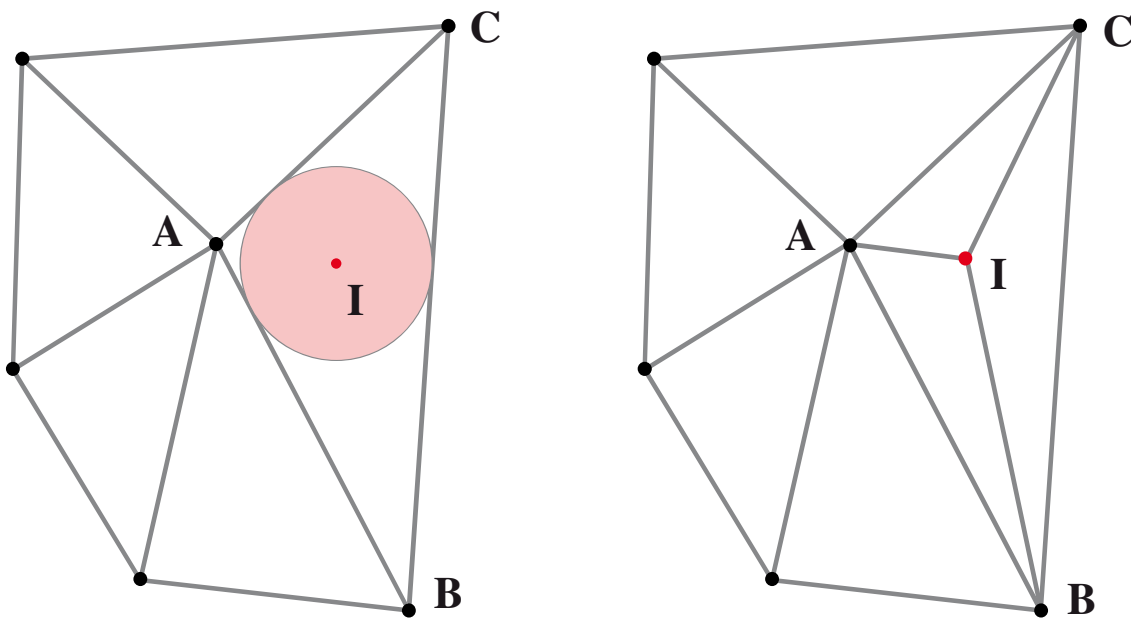


Figura 1: Inserción de un nuevo punto con una partición simple

Inserción de incentro del triángulo ABC, particionando el antiguo triángulo en tres nuevos triángulos ABI, BCI y CAI

Para evitar mallas con triángulos indeseados, los algoritmos de refinamiento pueden usar las siguientes medidas de calidad:

(i) Evitar los triángulos con una alta razón de aspecto $R_a = \frac{circunradio}{2 * inradio}$

(ii) Evitar triángulos con ángulos interiores pequeños

Se considerará un **triángulo de mala calidad**, como un triángulo presente en la triangulación que se desea refinar, de acuerdo a algún criterio (Figura 2).

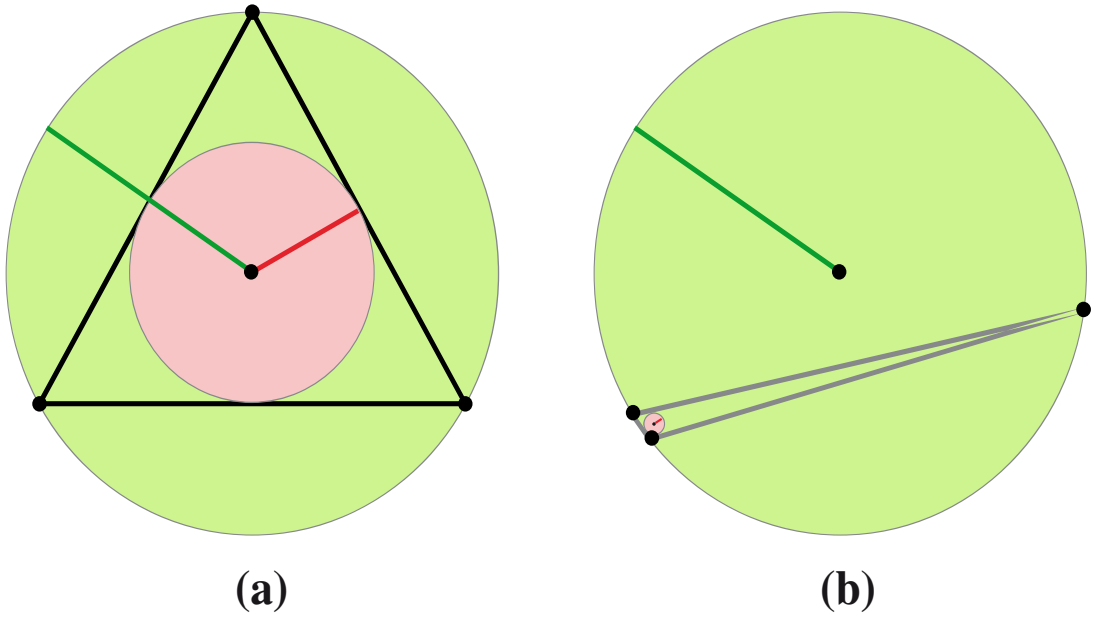


Figura 2: Calidad en triángulo
(a) Triángulo con razón de aspecto 1 y ángulos interiores iguales a 60º
(b) Triángulo con razón de aspecto 5 y ángulo mínimo interior 1º. Ejemplo clásico de triángulo de mala calidad

2.3. Triangulación Delaunay

Se define como **test del circuncírculo** al procedimiento que indica si dado un triángulo T y un punto p, p está fuera del circuncírculo de T. En caso afirmativo, se dirá que T no aprueba el test del circuncírculo. En caso contrario, sí lo aprueba.

Una triangulación se dice **triangulación Delaunay** si cada triángulo aprueba el test del circuncírculo con todos los vértices presentes en la triangulación. (Figura 3).

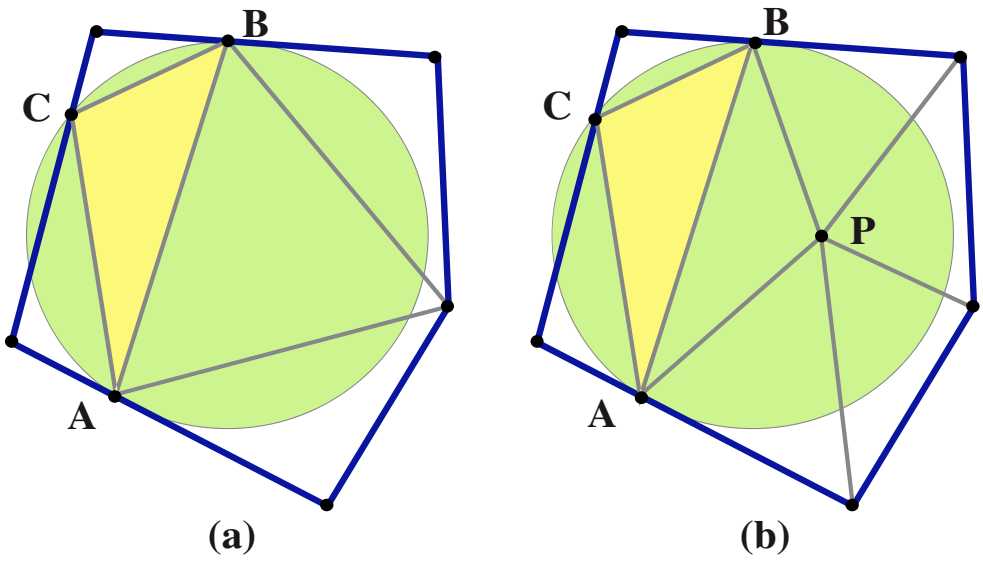


Figura 3: Triangulación Delaunay
(a) El triángulo ABC es un triángulo Delaunay.
(b) El triángulo ABC no aprueba el test del círculo con el vértice D, por lo que no es Delaunay

Esta condición maximiza el ángulo mínimo de todos los triángulos de la malla, lo que en sí es un gran avance, sin embargo, esto no necesariamente genera triángulos de una calidad esperada, por lo que los algoritmos de refinamiento realizan inserciones sucesivas de nuevos puntos a la malla, con la intención de que la inserción de estos nuevos puntos destruya los triángulos de mala calidad y se generen nuevos triángulos con mejores ángulos interiores. Cuando la inserción de un nuevo punto a la malla se preocupa de mantener la condición Delaunay, será denominada Inserción Delaunay.

2.3.1. Triangulación Delaunay Restringida

Una **triangulación restringida** es una triangulación en la cual cada arista restringida está presente en la triangulación en forma íntegra como uno y sólo un segmento de recta (el cual puede estar compartido). Esto da cabida a la existencia de triángulos que no aprueben el test del círculo.

Se define el **test del circuncírculo restringido**, como el test del circuncírculo con la siguiente modificación: un triángulo T que no cumple la condición Delaunay aprueba el test siempre y cuándo exista una arista restringida o un borde de malla que actúe como una muralla que bloquea la visión al resto de la malla, y por ende, a algún punto que esté en el interior del circuncírculo.

Se define una **triangulación Delaunay restringida** como una triangulación en la que todos sus triángulos aprueban el test del circuncírculo restringido. En la Figura 4 se muestran dos triangulaciones Delaunay restringidas

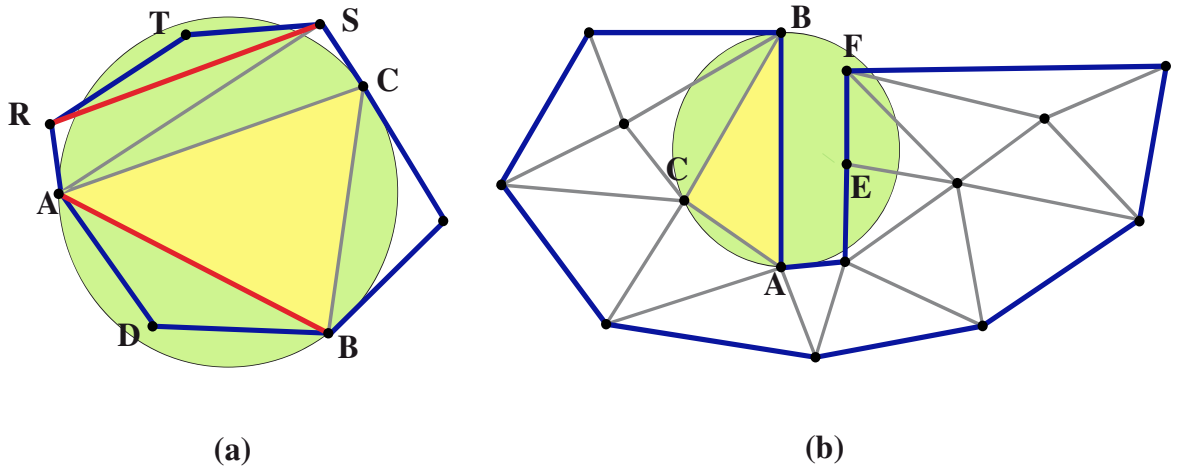


Figura 4: Triangulaciones Delaunay Restringidas

En ambas figuras, los segmentos rojos, azules y los vértices negros son elementos iniciales. (a) Las aristas restringidas RS y AB en la figura permiten ignorar los vértices T y D respectivamente. (b) El borde de malla AB permite ignorar los vértices E y F.

2.3.2. Triangulación Delaunay Conforme

Una **triangulación Delaunay conforme** es una triangulación Delaunay en la cual sus aristas restringidas están presentes en la triangulación como la unión de uno o más segmentos de recta (Figura 5).

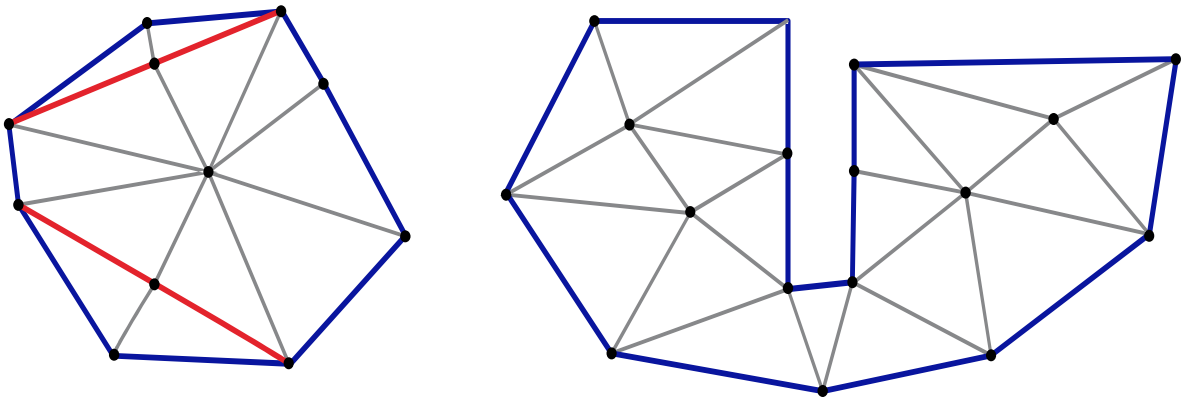


Figura 5: Triangulaciones Delaunay conformes
Las aristas restringidas y bordes de malla son los mismos que en la Figura 4, salvo que en este caso se encuentra divididas en aristas de menor tamaño.

2.4. Inserciones Delaunay

2.4.1. Inserción con intercambio de diagonales

Este algoritmo ([Law72]) comienza con la búsqueda del triángulo que incluye en su interior al nuevo punto:

- Si el nuevo punto se ubica en el interior de un triángulo, éste se debe reemplazar por los tres triángulos formados al particionar el triángulo con los segmentos que unen cada vértice con el nuevo punto. En la Figura 1 se muestra un ejemplo de partición simple de un triángulo debido a la inserción de un punto al interior de éste.

- Si el nuevo punto se ubica sobre la arista del triángulo, se debe considerar el vecino con quien se comparte dicho borde, particionando ambos triángulos con los segmentos que

unen el vértice opuesto a la arista y el nuevo punto (Figura 6)

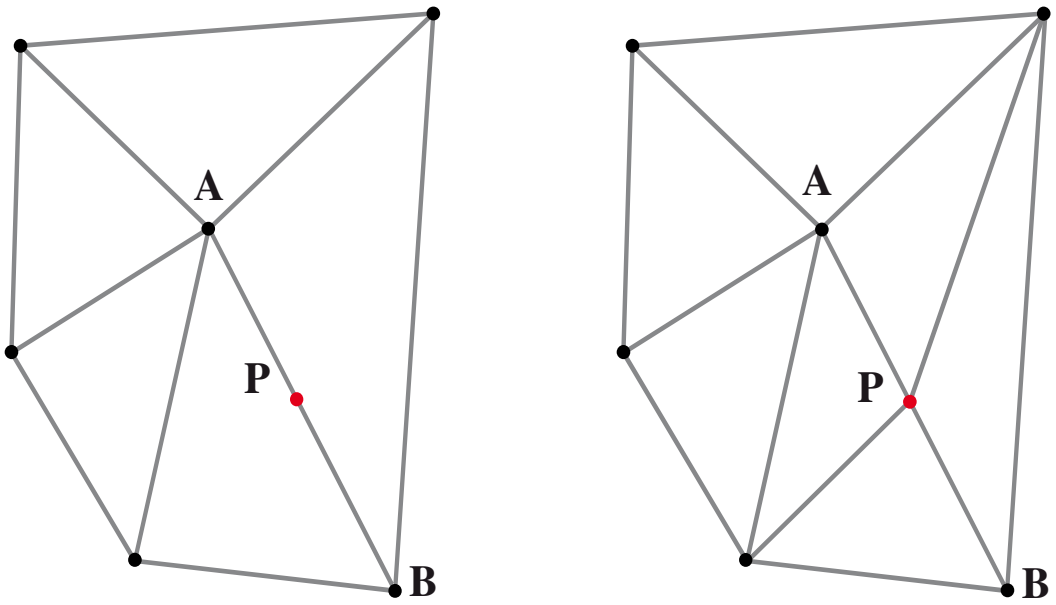
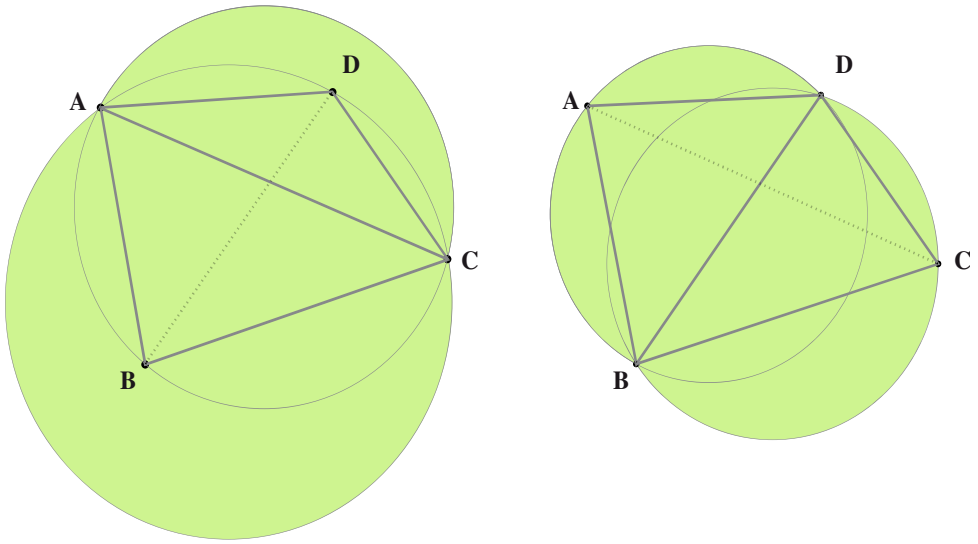


Figura 6: *Partición simple por inserción de nuevo punto que cae en el borde de un triángulo*

En seguida, se realiza un proceso de "Delaunización": por cada nuevo triángulo en la malla se realizará el test del circuncírculo con los vértices de sus vecinos. En caso de no aprobar el test, se realiza el "intercambio de diagonales" con el vecino, haciendo alusión al cuadrilátero formado por ambos triángulos (Figura 7). Los nuevos triángulos necesariamente cumplen la condición Delaunay, por lo que basta revisar recursivamente la condición Delaunay con los nuevos triángulos creados.



(a) (b)

Figura 7: Intercambio de diagonales

(a) El triangulo ABC no pasa el test del circuncírculo debido a vértice D, así como el triángulo ACD con el vértice B, por lo que se "intercambia la diagonal" y se reemplazan ambos triángulos por dos nuevos que si cumplen entre el test del circuncírculo (b)

2.4.2. Inserción por cavidad

Este algoritmo ([Wat81]) consiste en encontrar todos los triángulos que no aprueban el test del circuncírculo con el nuevo punto que se desea insertar (Figura 8a). Todo estos triángulos se deben eliminar de la triangulación, dejando una cavidad o hueco en la triangulación (Figura 8b), y se reemplazan con los triángulos resultantes de unir cada vértice de la cavidad con el nuevo punto (Figura 8c).

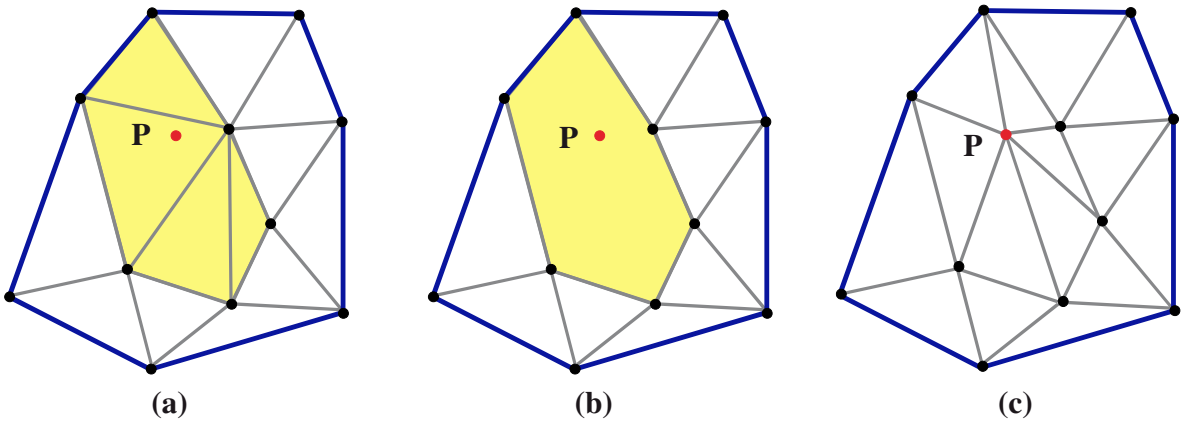


Figura 8: *Cavidad*

(a) Triángulos que no aprueban el test del círculo con el nuevo punto P (b) Cavidad resultantes (c) Inserción del nuevo punto en la cavidad

2.5. Algoritmos de Refinamiento

Los algoritmos de refinamientos investigados se han clasificado en dos tipos:

- **Refinamientos Delaunay**

  Si la triangulación inicial es una triangulación Delaunay, la triangulación resultante sigue siendo una triangulación Delaunay o triangulación Delaunay conforme, gracias a que los nuevos puntos de la malla se agregaron mediante inserciones Delaunay.

- **Refinamientos No Delaunay**

  Estos algoritmos no necesariamente mantienen la condición Delaunay si se contara con ella, ya que el algoritmo de inserción utilizado no se preocupa de mantener dicha condición. También se pueden aplicar en triangulaciones no Delaunay.

2.5.1. Refinamientos Delaunay

2.5.1.1. Algoritmo Circuncentro - Delaunay

Se define **circunferencia diametral** asociada a una arista, a la menor circunferencia que contenga completamente a dicha arista (Figura 9).

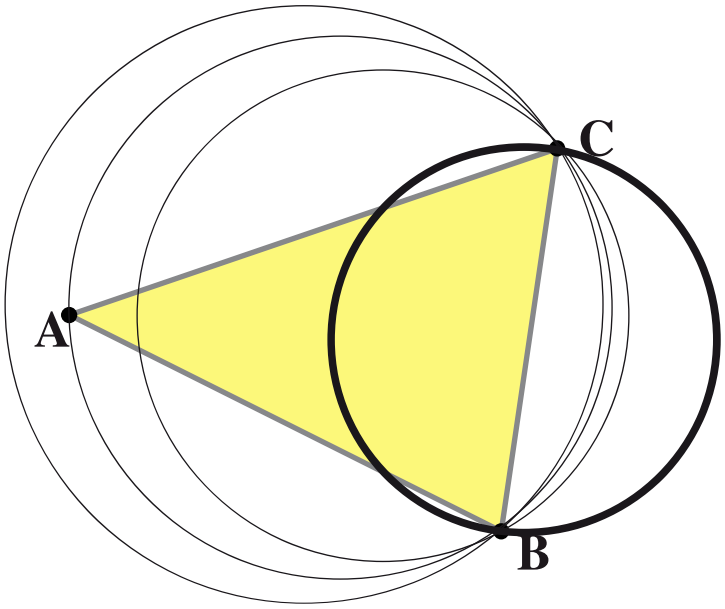


Figura 9: Circunferencia Diametral

Sólo una circunferencia, la de menor tamaño, contiene completamente a la arista AB como su propio diametro

Se define **arista encroached** como un borde de malla o arista restringida que contenga en el interior de su circunferencia diametral a algún vértice de la triangulación (Figura 10).

El algoritmo ([Rup95]) realiza sucesivamente dos pasos: un proceso pre-inserción y a continuación el refinamiento de un triángulo de mala calidad.

El proceso pre-inserción busca independizarse de todas los bordes de malla o aristas restringidas encroached. Para esto, se realiza un reemplazo consistente en dos nuevas aristas correspondientes

a la división de la arista original a través de su punto medio con una inserción Delaunay. El proceso finaliza cuando no hay ningún borde de malla o aristas restringidas encroached. Es importante destacar que este proceso no se realiza en base a la calidad de la malla, sin embargo, con la realización de éste, el algoritmo asegura que al momento de refinar algún triángulo de mala calidad, el nuevo punto caerá dentro del dominio que cubre la malla.

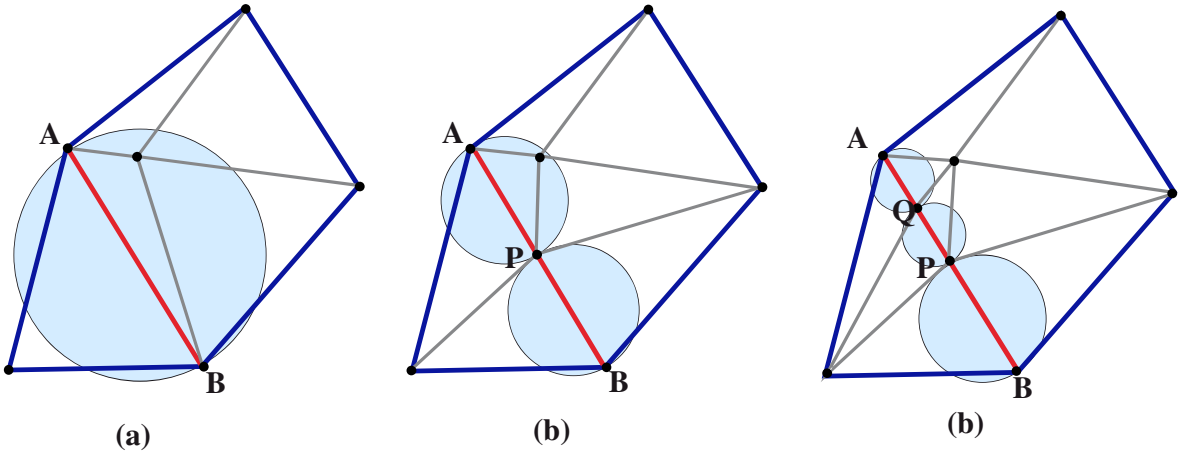


Figura 10: Eliminación arista encroached

(a) La circunferencia diametral de AB contiene en su interior un vértice. (b) Se inserta su punto medio P. (c) La nueva arista restringida AP cumple la misma condición, por lo que se inserta su punto medio Q

A continuación se procede a refinar un triángulo de mala calidad. Con el triángulo de mala calidad seleccionado, se calcula su circuncentro y se verifica si la inserción del circuncentro provocaría la existencia de un borde de malla o arista restringida involucrada: en caso de ser así, el punto que se inserta será el punto medio de la arista involucrada, y en caso contrario, se procede a insertar el circuncentro del triángulo (Figura 11).

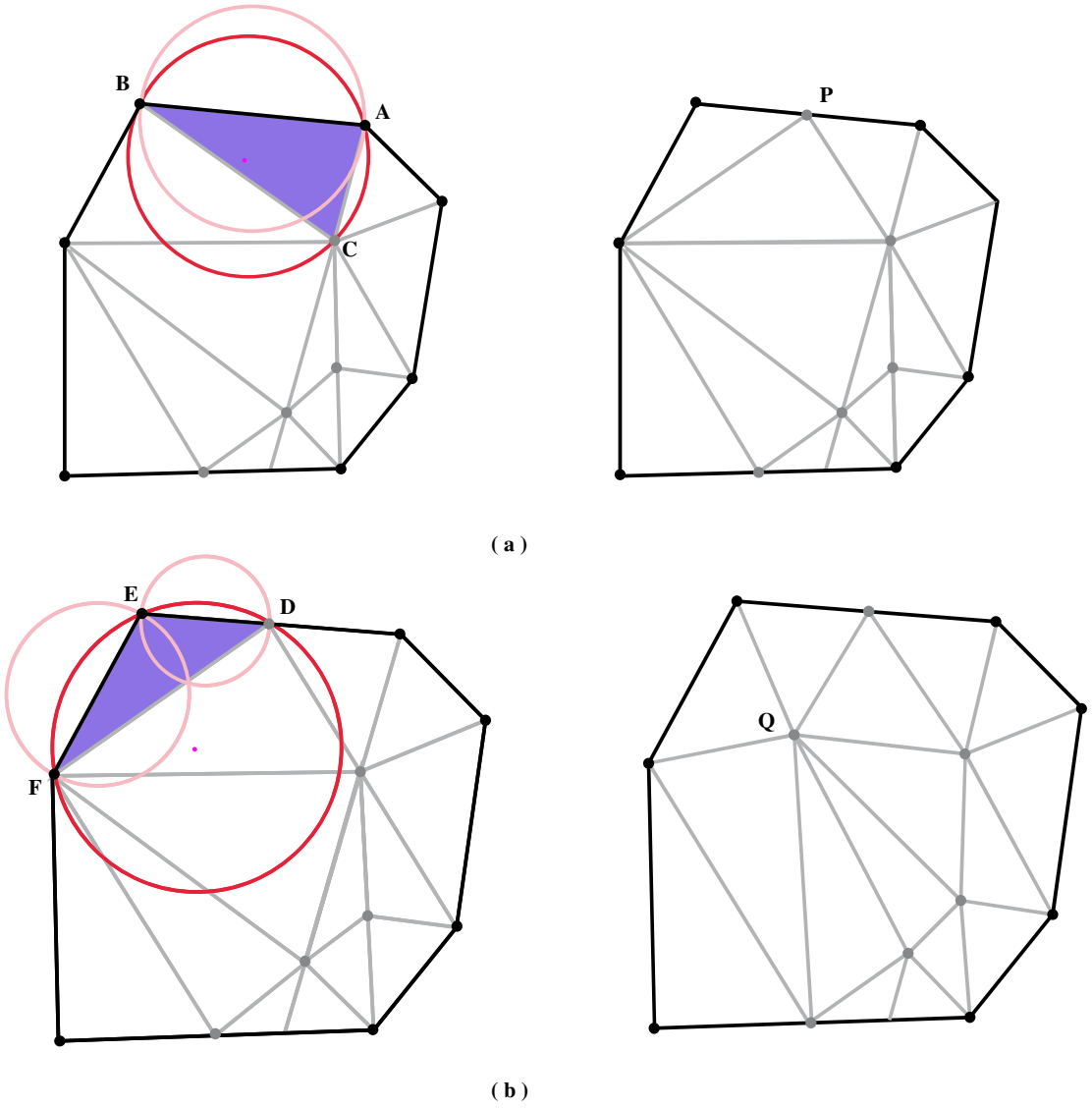


Figura 11: Refinamiento en Circuncentro - Delaunay

(a) El circuncentro del triángulo ABC involucra al borde de malla AB, por lo que se inserta el punto medio P de AB

(b) El circuncentro de DEF no involucra ninguna arista restringida, por lo que se inserta el circuncentro Q

2.5.1.2. Algoritmo Lepp - Punto Medio - Delaunay

Se define como **Lepp asociado a un triángulo** t_0 (del inglés Longest edge propagation path), como una lista ordenada de triángulos adyacentes $t_{0,}t_{1,}\ldots,t_n$ en la que se cumple que todo

triángulo es vecino de su predecesor a través de la arista más larga del vecino, y es vecino del sucesor a través de su propia arista más larga (Figura 12).

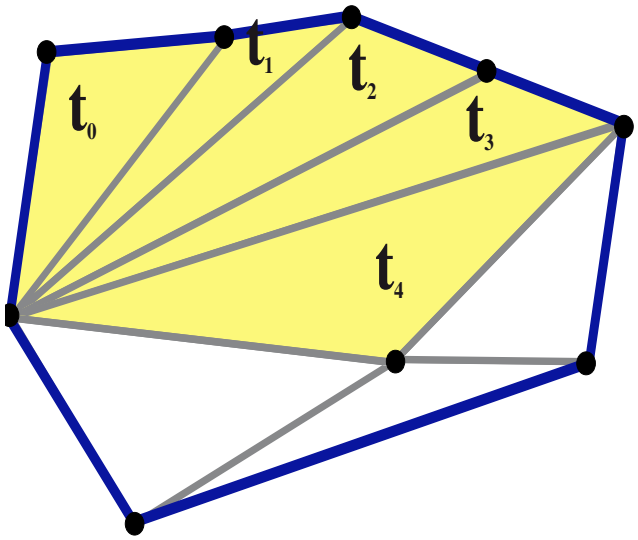


Figura 12: Lepp

El lepp del triángulo t_0 está formado por la sucesión de triángulos t_0, t_1, t_2, t_3 y t_4. La arista terminal es la arista que comparten t_3 y t_4.

Se define **arista terminal del Lepp asociado a un triángulo** t_0, a la arista más larga del triángulo t_n del Lepp (el último triángulo de la lista ordenada).

El algoritmo Lepp - Punto Medio - Delaunay ([Azo10]) fija el triángulo que se desea refinar y realiza sucesivamente una inserción Delaunay del punto medio de la arista terminal de dicho triángulo. El proceso se repite hasta que el triángulo por refinar ya no forma parte de la malla, es decir, fue reemplazado por un conjunto de nuevos triángulos (Figura 13).

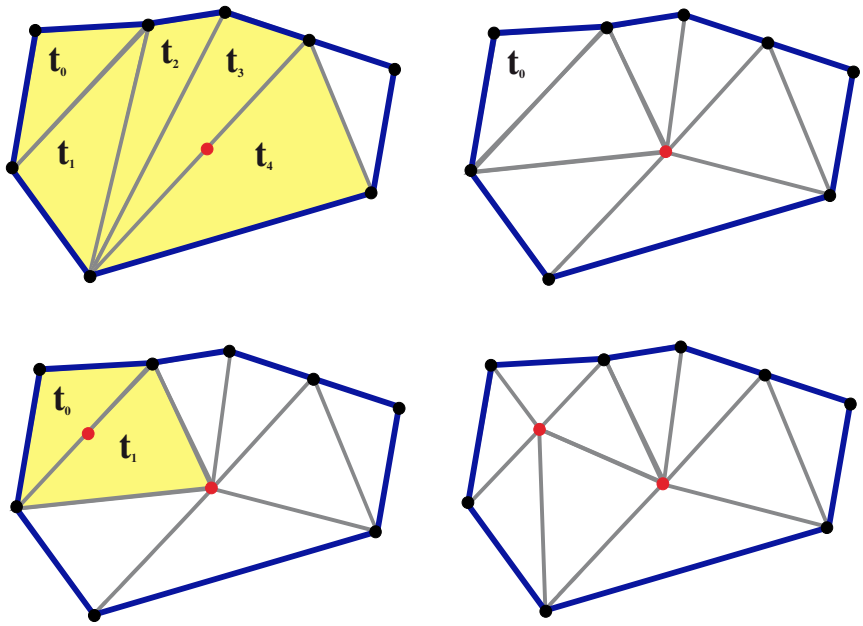


Figura 13: Refinamiento en Lepp - Punto Medio - Delaunay

Se realiza sucesivamente la inserción de la arista terminal del lepp del triangulo t_0

2.5.1.3. Algoritmo Lepp - Centroide - Delaunay

Este algoritmo ([Riv97][Riv10]) realiza un proceso idéntico al algoritmo Lepp - Punto Medio - Delaunay, sin embargo, la selección del nuevo punto que se inserta es distinta. En este caso, el nuevo punto seleccionado corresponde al centroide del cuadrilátero formado por los dos triángulos que comparten la arista terminal del Lepp (Figura 14). En caso de que la arista terminal sea un borde de malla (es decir, sólo un triángulo posee la arista terminal), el nuevo punto corresponde al punto medio de la arista terminal.

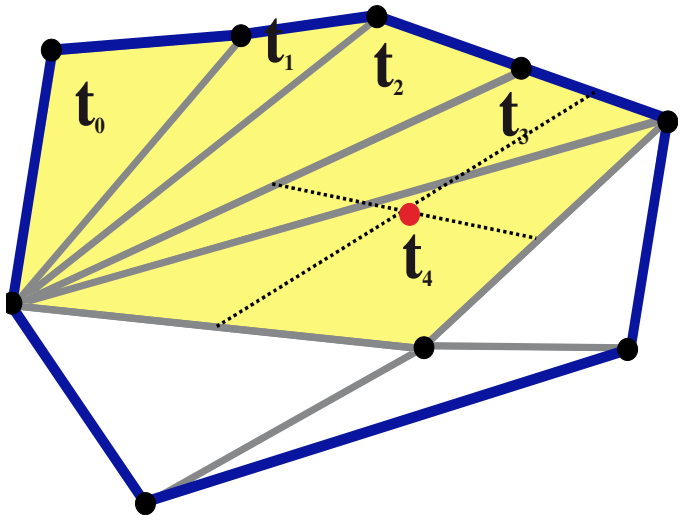


Figura 14: Lepp - Centroide - Delaunay

El centroide del cuadrilátero formado por los triángulos t_3 y t_4 se puede calcular como la intersección de los segmentos que unen los puntos medios de los lados opuestos.

2.5.1.4. Off-Center - Delaunay

El algoritmo Off-Center - Delaunay ([Ung04]) propone una alternativa a la elección del circuncentro como nuevo punto a insertar que establece el algoritmo de Circuncentro - Delaunay, aprovechando que en ese momento se conoce el ángulo umbral de calidad α.

Se define **off-center** como un punto ubicado en el segmento de recta que une el circuncentro de un triángulo y el punto medio de la arista de menor tamaño del triángulo (Figura 15). El objetivo del off-center es que el triángulo formado por la arista de menor tamaño del triángulo y el mismo off-center, tenga todos sus ángulos interiores mayores a una cota inferior α. El cálculo del off-center es necesario sólo si el circuncentro no cumple dicho objetivo.

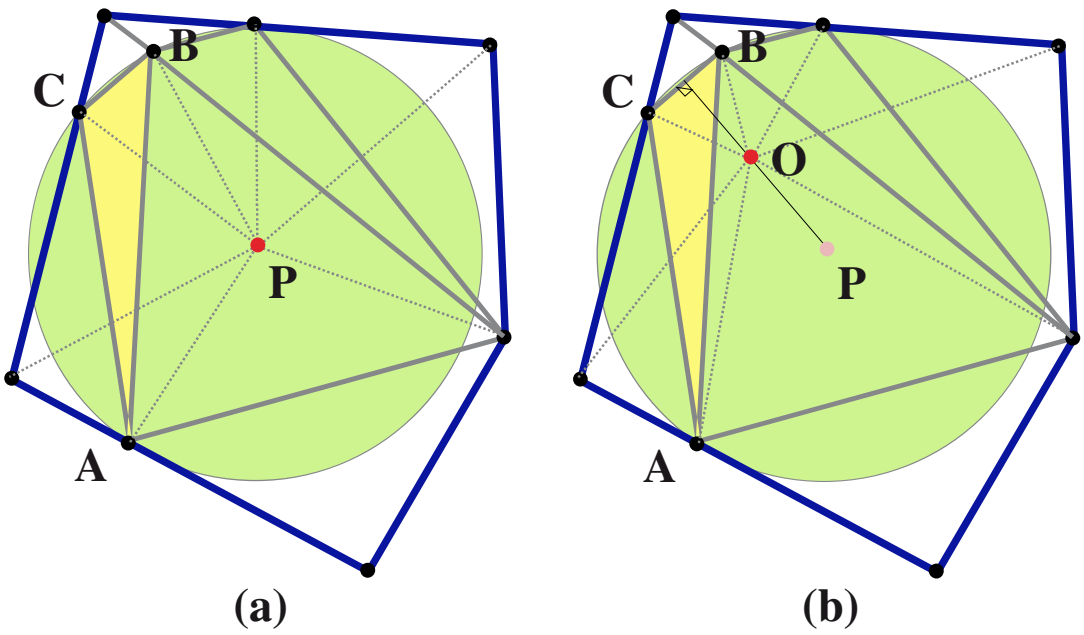


Figura 15: Off-Center

(a) La línea punteada muestra los triángulos que se formaría con la inserción del circuncentro P del triángulo ABC cuyo ángulo en A no cumple el límite α y CB es su arista de menor longitud. (b) El off-center se sitúa sobre el segmento que une el punto medio del CB con P, cumpliéndose que el ángulo en O del triángulo COB es igual mínimo valor permitido

2.5.2. Refinamiento No Delaunay

2.5.2.1. Algoritmo Lepp-Bisección

Este algoritmo ([Riv97][Riv08]) realiza un proceso idéntico al algoritmo Lepp - Punto Medio - Delaunay, sin embargo, no realiza una inserción Delaunay del nuevo punto. La inserción sólo realiza una partición local de los triángulos terminales del Lepp en la malla. Se reemplazan los triángulos que comparten la arista terminal por los cuatro triángulos resultantes de unir todos los vértices de los triángulos involucrados (o dos nuevos triángulos, en caso de que la arista sea borde de malla).

2.6. Recuperación de aristas restringidas

Si la malla inicial tiene restricciones, consistentes en segmentos de aristas, éstas deben estar presentes en la malla resultante del refinamiento representadas como aristas de triángulo. Si bien los algoritmos de inserción Delaunay pueden modificarse para respetar las aristas restringidas, es interesante estudiar algoritmos de recuperación de aristas restringidas

2.6.1. Algoritmo de Recuperación de Arista por Uniones

Este algoritmo inserta el punto medio la arista restringida, dividiendo recursivamente el problema en las dos aristas resultantes de dicha división, hasta que la arista restringida inicial se encuentre completamente cubierta como la unión de las aristas resultantes de cada división realizada (Figura 16). El resultado es una triangulación Delaunay conforme. Para asegurar que este algoritmo funcione correctamente es necesario detectar las aristas restringidas que se intersectan entre si, y dividirlas en el punto de intersección

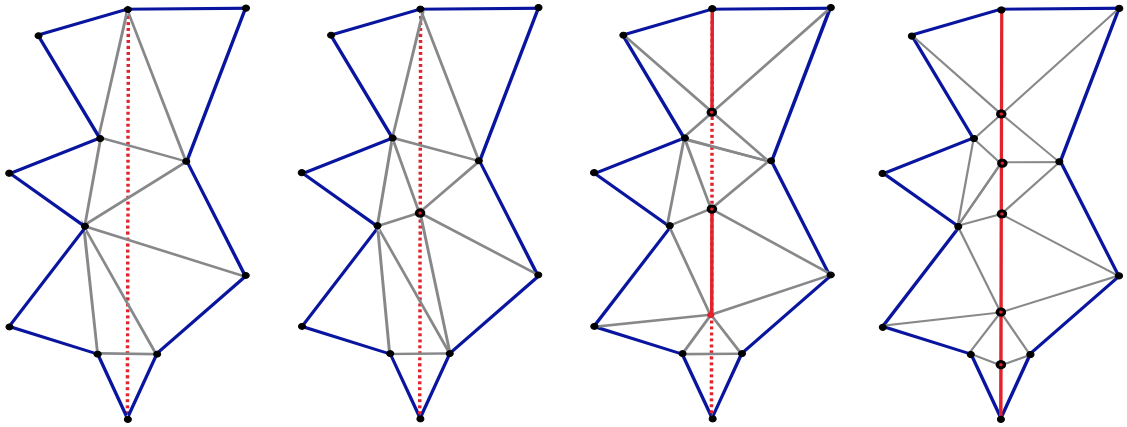


Figura 16: Recuperación de Aristas por Uniones

La arista restringida (línea punteada) se recupera finalmente como la unión de varias restringidas.

2.6.2. Algoritmo de Recuperación de arista completa

El algoritmo realiza una búsqueda de todos los triángulos que contienen en su interior (o borde) algún tramo de la arista restringida. En la cavidad resultante se procede a formar nuevos triángulos uniendo todos los puntos con uno de los vértices de la arista con el algoritmo propuesto en [Vig97]. El algoritmo se compone de los 3 siguientes pasos:

1. Encontrar los triángulos sobre los que cae la arista restringida y "reemplazarlos" por la arista restringida. Esto divide la cavidad en dos nuevas cavidades.

2. Por cada cavidad resultante, se debe encontrar el vértice en la cavidad tal que al formar un triángulo con los vértices de la arista restringida apruebe el test del circuncírculo con respecto al resto de los vértices de la cavidad e insertar dicho triángulo.

3. Al insertar un triángulo en la cavidad, ésta es dividida en dos cavidades nuevamente y basta realizar el proceso recursivamente en cada cavidad hasta que éstas sólo se encuentren conformadas por tres triángulos. El resultado es una triangulación Delaunay Restringida

En la Figura 17 se muestra el proceso de recuperación completa de una arista.

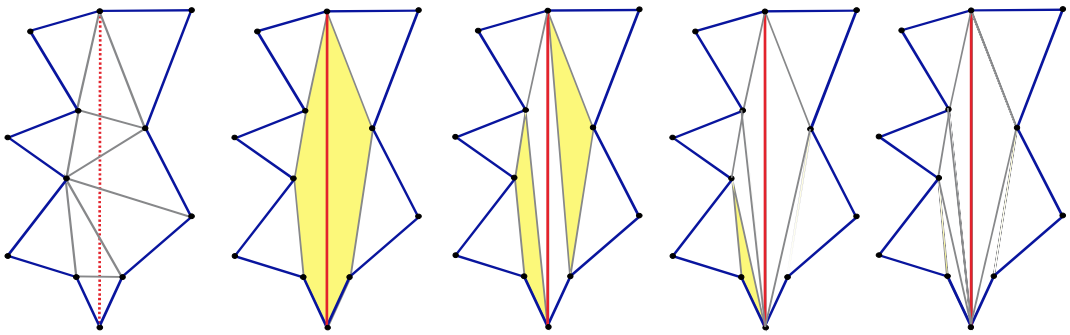


Figura 17: Recuperación de Arista Completa

La línea punteada representa la arista que se debe recuperar. El área coloreada muestra las cavidades que se deben triangularizar.

2.7. Selección de Triángulo para Refinar

Teniendo en cuenta que los algoritmos buscan mejorar la calidad de la malla, eliminando triángulos que no cumplan con un mínimo ángulo interior, el acercamiento natural consistiría en eliminar todos los triángulos, independiente del orden que se use. Más aún, en la definición de cada algoritmo no se hace explícita alguna forma en particular en el orden en que se deban elegir los triángulos para su refinamiento, y por consiguiente, eliminación de la malla. En [Ung04] se introduce el tema de la selección de triángulos de mala calidad y cómo una distinta selección puede afectar los resultados finales, ya sea en el número de nuevos puntos necesarios para refinar totalmente la malla y/o en el número de triángulos resultantes. Los criterios usados y mencionados en los algoritmos estudiados son los siguientes:

- Prioridad a triángulos con menor ángulo interior bajo el umbral
- Prioridad a triángulos con arista de menor tamaño y con al menos un ángulo interior bajo el umbral

Además, se debe considerar los algoritmos Lepp-Delaunay, Lepp-Centroide y Lepp-Bisección al momento de refinar un triángulo realizan sucesivas inserciones hasta destruir el triángulo seleccionado, independiente si las inserciones realizadas generaron algún triángulo con una peor calidad al triángulo objetivo.

CAPÍTULO 3

PROPUESTA DE MARCO DE EXPERIMENTACIÓN

La solución propuesta en esta memoria para cumplir los objetivos propuestos ha sido desarrollada en el lenguaje C++ con librerías Qt para el desarrollo de la interfaz gráfica y la librería OpenGL para la visualización de la malla.

La elección de C++ se debió principalmente a la eficiencia y rapidez que posee al ser un lenguaje compilado, además de ser un lenguaje orientado a objetos.

La elección de Qt para el desarrollo de la interfaz gráfica se basó en la alta madurez de la librería, la cual además dispone para los desarrolladores de herramientas como QT Designer para diseñar y generar interfaces gráficas con herramientas de algo nivel y QT Creator que permite llevar un proyecto C++ con todas las herramientas de compilación y *debugging* necesarias para un correcto desarrollo sin pérdidas de tiempo innecesarias.

3.1. Proceso de Refinamiento

En el caso de los algoritmos Delaunay, el problema de refinamiento puede traducirse a un

problema de optimización: producir una malla de mejor calidad con la menor cantidad de inserciones de nuevos puntos, sujeta a las restricciones geométricas de la malla inicial y al nivel de calidad requerido. Por lo tanto, es interesante estudiar los resultados que obtienen los algoritmos y cómo estos resultados varían cuando se realiza alguna modificación. En el caso de los algoritmos no Delaunay, como Lepp- Bisección, no hay control en el número de puntos insertados, por lo que en ese caso, el problema se concentra en no realizar más inserciones de las estrictamente necesarias.

Tomando como base los algoritmos estudiados, el proceso de refinamiento puede generalizarse en el siguiente conjunto de pasos:

(i) Proceso Pre-Inserción: dada una malla, se realizan modificaciones a ésta antes de realizar inserciones para mejorar su calidad.

(ii) Criterio de selección de triángulo para refinar: dada una malla, se selecciona el triángulo que se procederá a refinar.

(iii) Criterio de selección de nuevo punto para insertar: dado un triángulo, se define qué punto se procederá a insertar

(iv) Método de inserción de nuevo punto a la malla: dado un punto, se realiza la inserción en la malla

(v) Recuperación de aristas restringidas: dada una malla y un conjunto de aristas restringidas, se realizan un proceso de recuperación de aristas restringidas, en caso de que éstas no se encuentren en la malla.

De los cuales los pasos i, ii, iii y iv son repetidos mientras sea necesario refinar la malla.

Cada algoritmo de refinamiento estudiado puede descomponerse en estas cinco partes, que representan decisiones que se toman en determinados momentos del algoritmo. Cabe destacar que

si bien estas decisiones cubren a todos los algoritmos, no todas son obligatorias para un algoritmo. Por ejemplo, de todos los algoritmos estudiados, el algoritmo de Circuncentro - Delaunay y el algoritmo Off-Center - Delaunay son los únicos que realizan un paso pre-inserción (eliminar las aristas involucradas) y ninguno de los algoritmos Lepp tiene un proceso de ese tipo (sin embargo, podría usarse si esto permite que se inserten menos puntos a la malla). En la Figura 18 se presenta un diagrama que resume el proceso de refinamiento propuesto.

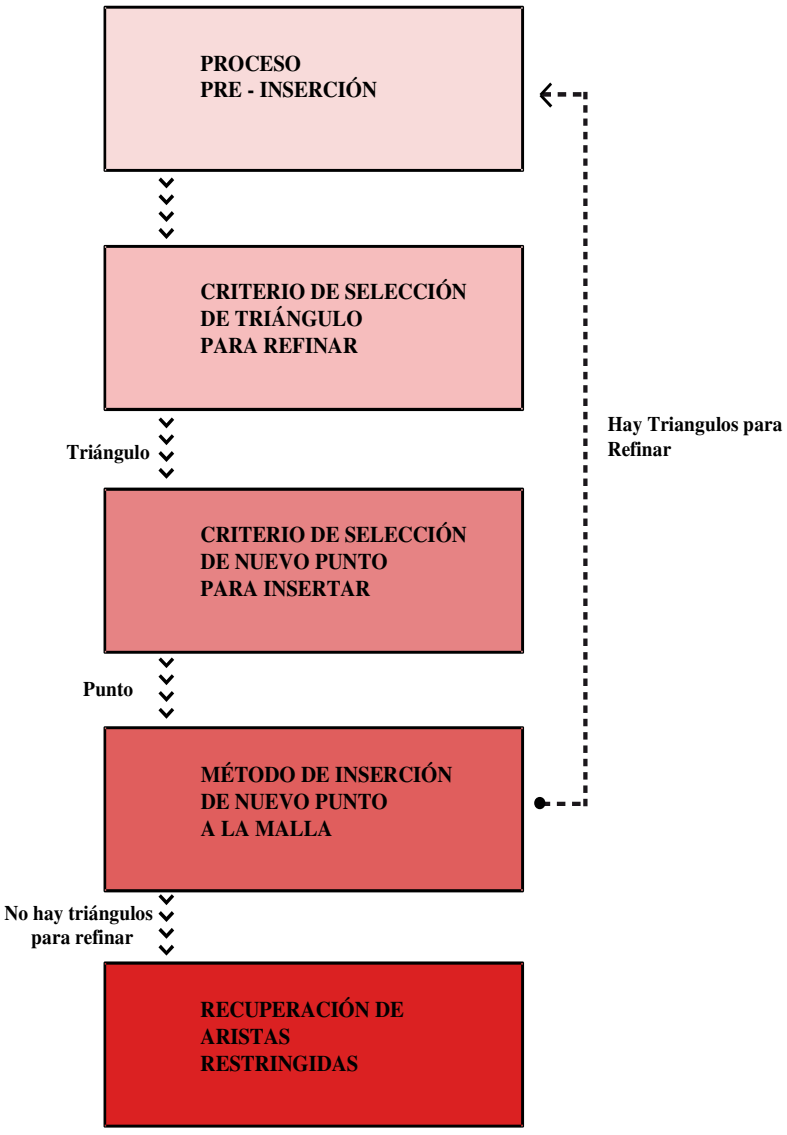


Figura 18: Proceso de refinamiento propuesto

3.2. Estructuras de Datos

Si bien existen una amplia investigación sobre estructuras de datos para representar mallas, como lo son las estructuras *quad-edge* y *winged-edge*, se decidió utilizar estructuras de datos simples para representar los elementos topológicos que se deben modelar. El objetivo es que el manejo de mallas, triángulos y vértices no agregue complejidad innecesaria al momento de modificar o extender la aplicación con nuevos algoritmos.

3.2.1. Point y Vertex

Se considerarán como el elemento geométrico punto (objeto Point) cualquier par ordenado (x, y) que no esté presente en la malla. Se considerará el elemento topológico vértice (objeto Vertex) el cuál corresponde a un objeto Point extendido con un identificador.

De esta manera, los atributos de esta clase son:

```
class Point{
protected:
      double xp;
      double yp;
public:
      Point(double x, double y);
      double x();
      double y();
      void setX(double x);
      void setY(double y);
      void glDraw();
}

class Vertex: public Point{
private:
      int idp;
public:
    Vertex(int id, double _x, double _y);
    int id();
    void glDraw();
    ~Vertex();
}
```

3.2.2. Triangle

Un triángulo (objeto Triangle) está compuesto por los siguientes elementos:

- un identificador
- referencias a sus tres vértices en sentido CCW: v_0 , v_1 y v_2
- referencias a sus tres triángulos vecinos : t_0 , t_1 y t_2 , los cuales son, respectivamente, los triángulos opuestos a los vértices v_0 , v_1 y v_2
- indicadores para saber si una arista es restringida, las cuales siguen la misma convención que los triángulos vecinos.

De esta manera, los atributos de esta clase son:

```
class Triangle{
private:
  int idp;
  QVector<Vertex*> vertexsp;
  QVector<Triangle*> neighboursp;
  QVector<bool> restrictedEdgesp;
public:
    Triangle(int id, Vertex* v0, Vertex* v1, Vertex* v2);
    int id();
    Triangle* getNeighbour(Vertex* P);
    Triangle* getNeighbour(int p);
    int isNeighbour(Triangle* neighbour);
    Vertex* getVertex(int p);
    void glDraw(Constant::GLTriangleType type, double value);
    Constant::IncludeCase include(Point* point);
    Constant::IncludeCase circumcircleInclude(Point* p);
    void setNeighbour(int pos, Triangle* neighbour);
    int getLongestEdge();
    double getLongestEdgeValue();
    int getSecondLongestEdge();
    bool isRestricted(int edge);
};
```

En la Figura 19 se presenta la convención fijada para la geometría de un triángulo.

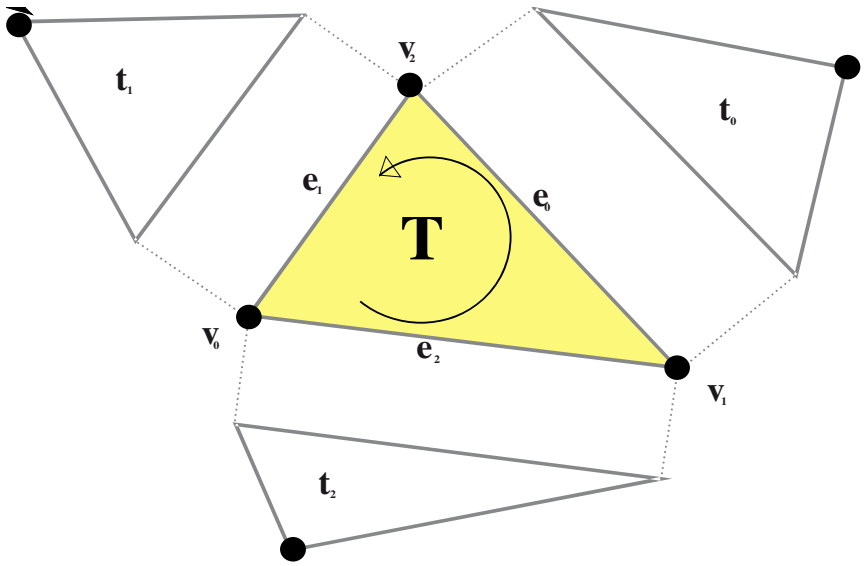


Figura 19: Convención en la nomenclatura de un triángulo

3.2.3. Mesh

Una malla (objeto Mesh) está compuesto de los siguientes elementos:

- Un map (id, Vertex)
- Un map (id, Triangle)
- Un map con las restricciones, donde cada restricción está formada por una colección ordenada de vértices
- Tres indicadores que almacenan el mayor identificador de vértices, mayor identificador de triángulos y mayor identificador de restricciones, con el objetivo de asignar nuevos identificadores para nuevos triángulos que se agreguen a la malla y así evitar colisiones de identificadores

- Características de la malla como los mínimos y máximos valores en cada uno de los ejes y el centro de la malla

```
class Mesh{
private:
  QHash<int, Vertex*>   vertexsp;
  QHash<int, Triangle*>  trianglesp;
  QHash<int, QVector<Vertex*> > restrictionsp;
  int cv, ct, cr;
  int lowerX, higherX, lowerY, higherY;
  Point* centerp;
public:
    Mesh(QString fileName);
    Triangle* createAndAddTriangle(Vertex* A, Vertex* B, Vertex *C);
    QVector<Vertex*> createAndAddRestriction(Vertex* A, Vertex* B);
    Vertex* createAndAddVertex(double x, double y);
    void drawMesh();
    Triangle* getTriangle(Point* p);
    void removeTriangle(Triangle* T);
    void removeAndDeleteTriangle(Triangle* T);
    int vertexsSize();
    int trianglesSize();
};
```

3.3. Diseño de Clases

Uno de los principales objetivos de esta memoria es desarrollar una solución que permita experimentar con las técnicas mencionadas en la sección anterior, de una manera simple y elegante, sin entrar en grandes modificaciones de la aplicación, ya sea al momento de desear una nueva combinación de técnicas o al momento de extender la aplicación con nuevos algoritmos.

Para solucionar esto se utilizaron patrones de diseño orientados a objetos, los cuales permiten atacar los problemas con soluciones efectivas y probadas.

3.3.1. Patrón Estrategia en pasos de Proceso de Refinamiento

El objetivo buscado con el uso de este patrón es que el proceso de refinamiento pueda ejecutarse

llamando al método estático *process* de la clase singleton RefinementProcess, en código C++:

```
class RefinementProcess{
public:
     RefinementProcess instance = 0;
     RefinementProcess *getInstance();
     static void process(Mesh m){
          Context* context = new
Context(GUI.getInstance().getOptions());
          context.pre_process(m);
          Triangle* t = context.select_triangle(m, true);
          while (t != null){
               Configuration* c = context.select_new_point(m, t);
               context.insert_point(m, c);
               t = context.select_triangle(m);
          }
          context.fix_restrictions(m);
     }
private:
     RefinementProcess(){};
}
```

En el proceso anterior, es el contexto (objeto Context) el que realiza en forma transparente cada paso del proceso haciendo uso de *interfaces*:

```
class Context{
private:
     PreProcessInterface ppi;
     TriangleSelectionInterface tsi;
     NewPointSelectionInterface npsi;
     PointInsertionInterface pii;
     RestrictionAdjustmentInterface rai;
     double p;
     Triangle* lastSelected;

public:
     Context(Options *o);
     void pre_process(Mesh* m){
          ppi.execute(m);
     }
     Triangle* select_triangle(Mesh* m, Bool last){
          if(this->lastSelected != null && last )
               return this->lastSelected;
          last = tsi.execute(m, p);
          return last;
     }
     Configuration* select_new_point(Mesh m*, Triangle* t){
```

```
            return npsi.execute(m, t);
        }
        void insert_point(Mesh m, Configuration* c)
            pii.execute(m, c);
        }
        void fix_restrictions(Mesm m){
            rai.execute(m);
        }
}
```

En esta implementación, la clase *Context* reúne el contexto de cinco patrones estrategia, uno por cada paso del proceso de refinamiento. El cliente en este caso es la clase *RefinementProcess*, la cual por medio del constructor de *Context*, configura el contexto con las estrategias seleccionadas por la interfaz gráfica.

De esta forma, para agregar un nuevo algoritmo en cualquiera de los pasos del proceso de refinamiento, sólo basta extender la *interface* que corresponda al paso en cuestión y asignar dicho objeto al contexto al momento de iniciar el proceso:

- Toda implementación de `PreProcessInterface.h` debe implementar un método `execute` que reciba como parámetro un objeto Mesh y que realice sobre esa malla los cambios necesarios del pro-proceso que se desea implementar.

- Toda implementación de `TriangleSelectionInterface.h` debe implementar un método `execute` que reciba como parámetro objeto `Mesh` y un double para fijar el umbral de calidad al momento de buscar un nuevo triángulo para refinar. Este método no siempre es llamado, ya que existe la opción de que el contexto devuelva el último triángulo seleccionado para refinar, si así es solicitado.

- Toda implementación de `NewPointSelectionInterface.h` debe implementar un método `execute` que reciba como parámetro un objeto `Mesh` y uno `Triangle`, y debe retornar un objeto `Configuration`, el cual almacena la información necesaria para ubicar el o los nuevos puntos a insertar, además de indicar si dichos puntos se

encuentran sobre alguna arista del triángulo, y así evitar cálculos innecesarios en el futuro, ya que la precisión del flotante puede no permitir detectar esa condición en algún cálculo futuro.

```
class Configuration{
private:
	Triangle* targetTriangle;
	vector<Point*> new_points;
	vector<int> edges;
}
```

- Toda implementación de `PointInsertionInterface.h` debe tomar un objeto `Mesh` y uno `Configuration` y realizar la inserción en la malla de acuerdo a lo que indique la configuración

- Toda implementación de `RestrictionRecoveryInterface.h` debe implementar un método `execute` que reciba como parámetro un objeto Mesh y que realice sobre esa malla los cambios necesarios de la recuperación que se desea implementar.

En el diagrama de la Figura 20 se muestra un diagrama parcial de clases con el uso del patrón estrategia y las clases e interfaces creadas.

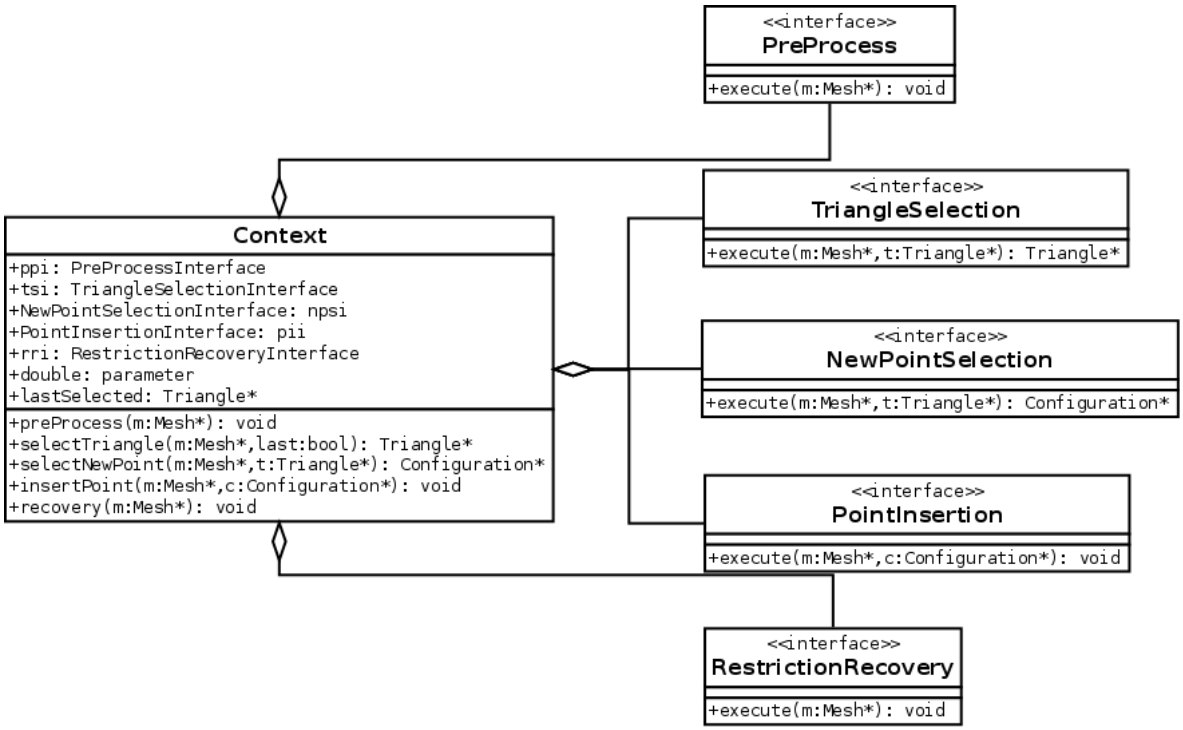


Figura 20: Diagrama de clases parcial con uso de patrón estrategia en el proceso de refinamiento.
Cada interfaz debe ser implementada con algún algoritmo concreto

3.3.2. Patrón Fábrica en extensión de nuevos algoritmos

Para que la extensión de la aplicación con nuevos algoritmos (en cualquiera de los pasos del proceso de refinamiento) se pueda realizar de una forma limpia y localizada, se utilizó el patrón fábrica, de modo que el ingreso de un nuevo algoritmo, se resuma en los siguientes pasos:

- Agregar en su fábrica correspondiente (objeto Factory) la opción que retorne el objeto correspondiente. El patrón fábrica también necesita que el objeto creado extienda de una *interface* en particular, sin embargo, ese aspecto ya se tiene gracias al patrón estrategia explicado anteriormente

- Enlazar la interfaz gráfica con el nuevo elemento de la fábrica

El diagrama de clases parcial de la Figura 21 muestra el uso del patrón fábrica en los pasos del proceso de refinamiento.

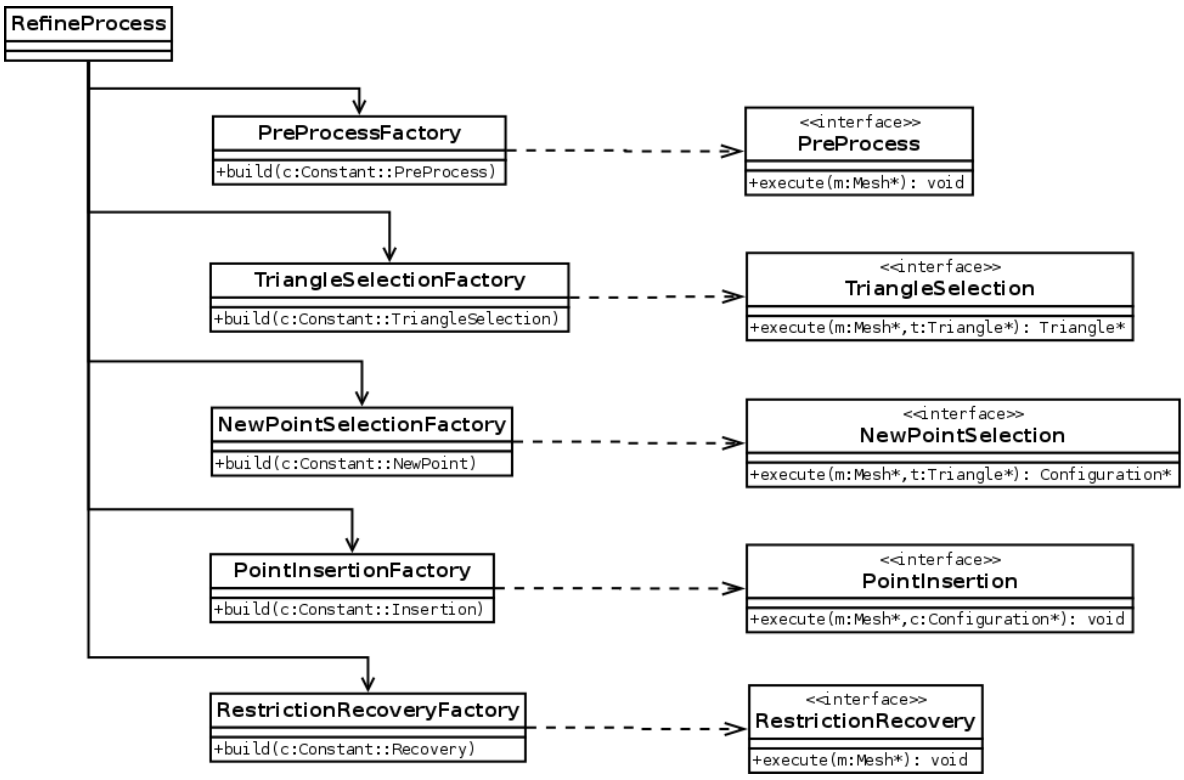


Figura 21: Diagrama de clases parcial con uso de patrón fábrica en la creación de algoritmos.

3.4. Sub-Algoritmos implementados

Para los 5 pasos del proceso de refinamiento se identificaron e implementaron los siguientes sub-algoritmos (separados por cada paso del proceso de refinamiento):

(i) Procesamiento pre-inserción

- Eliminación de bordes y aristas restringidas *encroached*

(ii) Selección de triángulo para refinar

- Sin priorización
- Prioridad a triángulos con menor ángulo bajo el umbral
- Prioridad a triángulos con arista de menor tamaño y con ángulo bajo el umbral

Además, se agrega la opción para seleccionar un mismo triángulo hasta que sea destruido.

(iii) Selección de nuevo punto para insertar a la malla

- Punto Medio Arista Terminal Lepp
- Circuncentro o Punto Medio de Arista Encroached por Circuncentro
- Centroide cuadrilátero Arista Terminal Lepp
- Off-Center o Punto Medio de Arista Encroached por Circuncentro

(iv) Inserción de nuevo punto

- Inserción Básica
- Inserción con Intercambio de Diagonales
- Inserción con Cavidad

(v) Recuperación de aristas restringidas

- Recuperación por Unión de Aristas

- Recuperación Completa

Esta separación en las implementaciones permite realizar cualquier combinación posible entre los cinco pasos. Si bien todas las combinaciones entre sí son compatibles en cuanto a la ejecución del proceso de refinamiento, no todas tienen sentido al combinarse (por ejemplo, insertar un off-center con una partición simple). Aún así, esta libertad puede llevar a realizar combinaciones que efectivamente mejoren el rendimiento de algún algoritmo en particular.

El diagrama de clases parcial de la Figura 22 muestra las clases necesarias para contar con estos algoritmos, exactamente, una extensión por cada algoritmo.

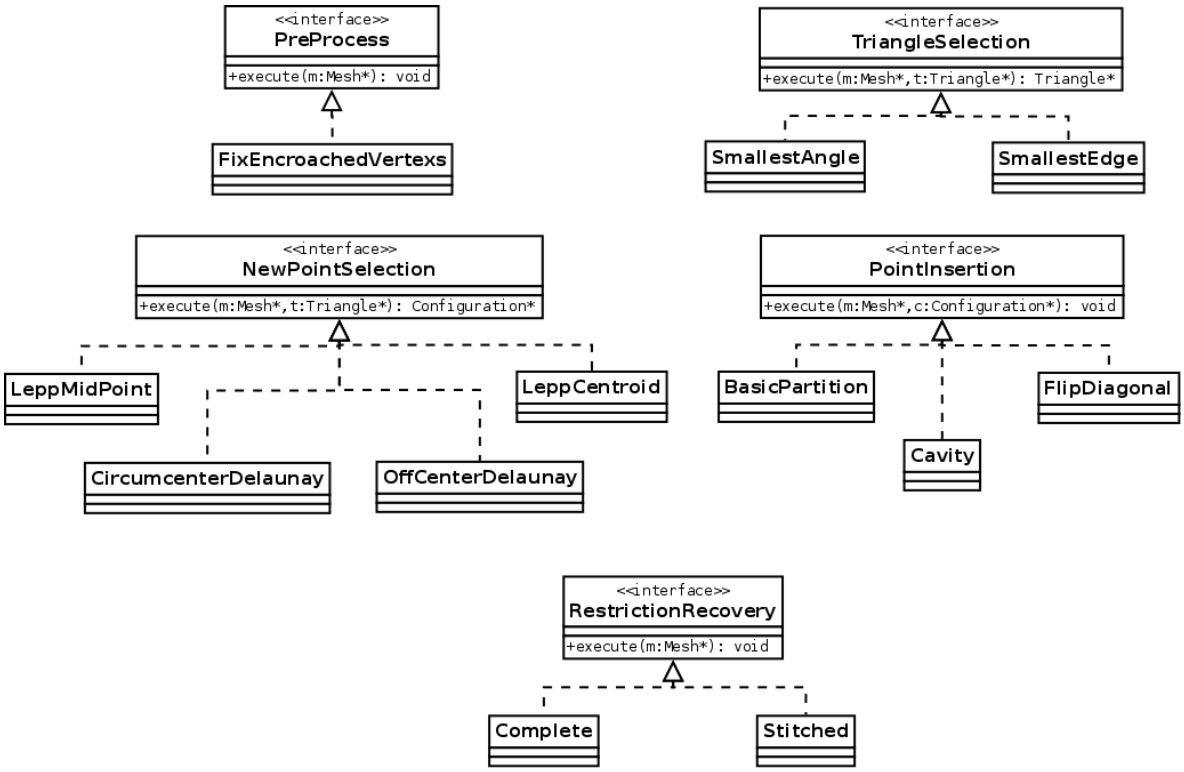


Figura 22: Diagrama de clases de algoritmos implementados.

3.5. Interfaz Gráfica

La creación y diagramación de la interfaz gráfica se realizó completamente con el framework Qt Creator, el cual provee todas las herramientas visuales necesarias para realizar una interfaz gráfica con la librería de desarrollo Qt. Además el framework aporta con librerías de gran utilidad:

- Widget de OpenGL para el dibujo de la malla
- Widget de interacción con el usuario, como Buttons, Text Inputs, DropBoxs, Checkboxs, entre otros
- Librerías geométricas y aritméticas

En la Figura 23 se puede apreciar la diagramación de la interfaz gráfica, la cual está dividida en cuatro áreas específicas. Un primer sector, denominado Visor, está dedica al dibujo del estado actual de la malla. Un segundo sector, Control, contiene los elementos de manejo del visor (zoom y posición) y del proceso (avanzar en refinamiento, refinamiento automático, detención). El tercer sector, Información muestra la información de la malla después de cada proceso de refinamiento. Un último sector, Configuración, contiene las opciones disponibles para cada paso del proceso de refinamiento, que consisten principalmente en los algoritmos implementados para caso paso. Los elementos seleccionados en Configuración determinan el contexto del proceso de refinamiento (objeto Context visto en 3.3.1).

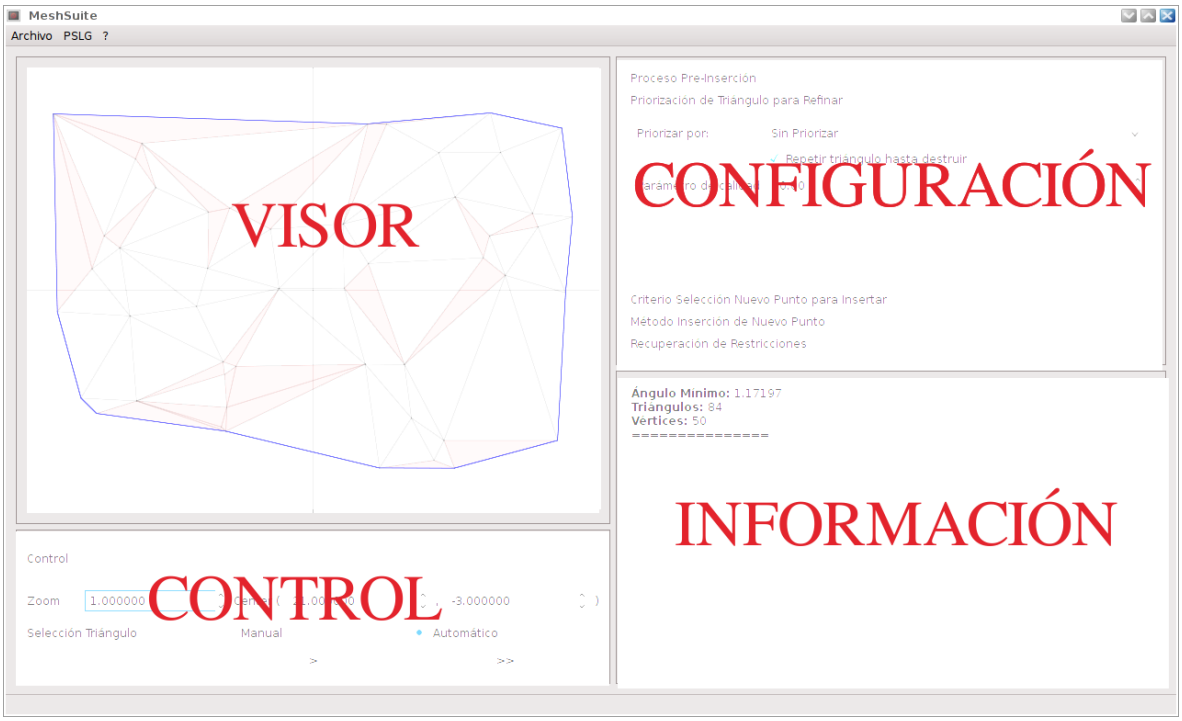


Figura 23: Diagramación Interfaz

En forma anexa se han agregado funcionalidades a la interfaz gráfica que permiten interactuar y obtener información en forma gráfica con la malla en proceso y habilitar una instancia de selección manual de triángulo para refinar mediante el uso del puntero. En este modo, el triángulo seleccionado es dibujado junto a su circuncírculo. Otra funcionalidad gráfica que permite guiar la experimentación es la coloración de los triángulos que ángulos pequeños, con un umbral que puede ser configurado en la interfaz (Figura 24).

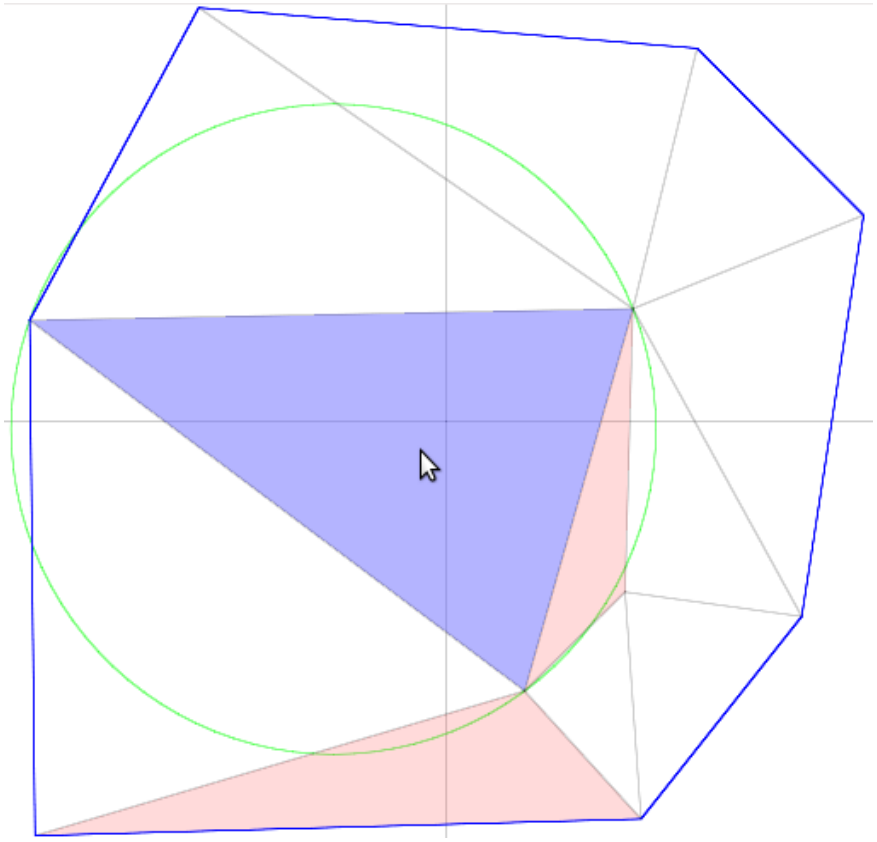


Figura 24: Ayudas gráficas

Las mallas dibujadas en el visor tienen marcados los triángulos de mala calidad. El triángulo seleccionado con el puntero se muestra de un color distinto y se dibuja su circuncírculo.

3.6. Ejemplo de Uso

3.6.1. Cargar o Generar Malla

El cargado de mallas archivos contenedores de PSLG con la siguiente estructura (extraída de Triangle):

```
<número de puntos>
<id> <coordenada 1> <coordenada 2>
...
<número de aristas restringidas>
<id> <punto inicio> <punto final> <tipo>
...
<número de huecos>
<id> <coordenada 1> <coordenada 2>
...
<número de triángulos>
<id> <vértice> <vértice> <vértice>
...
```

Si las aristas no están incluidas en el archivo la aplicación genera una triangulación Delaunay conforme con un algoritmo incremental. En la Figura 25 se muestra la vista de la aplicación al momento de cargar una malla.

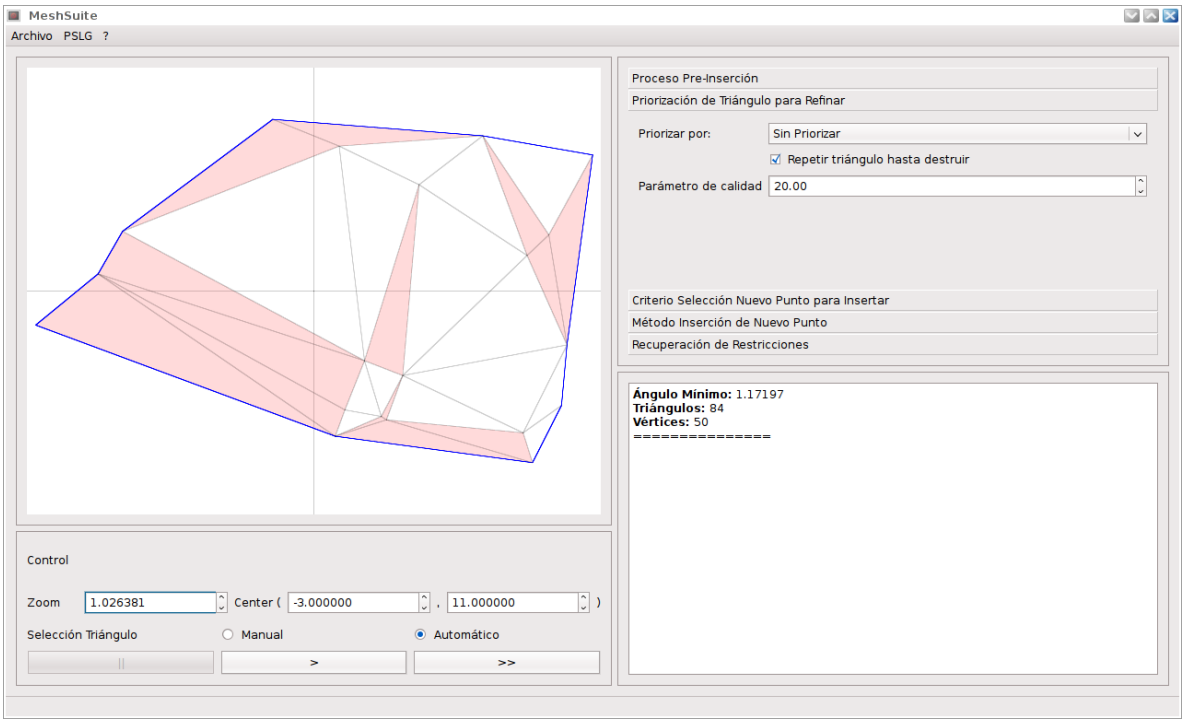


Figura 25: GUI Cargar PSLG

Vista de la interfaz luego de cargar una malla. El zoom y posición del centro se adaptan automáticamente en el Visor y Control, y se muestra la información inicial de la malla en Información

3.6.2. Configuración

En el sector de configuración es donde se configura los sub-algoritmos de cada parte del proceso de refinamiento que se desea utilizar. Para cada parte del proceso se debe seleccionar un algoritmo desde los menús desplegables y en el caso del algoritmo de selección de triángulos se debe entregar un parámetro que será utilizado por el algoritmo de selección para decidir qué triángulos refinar.

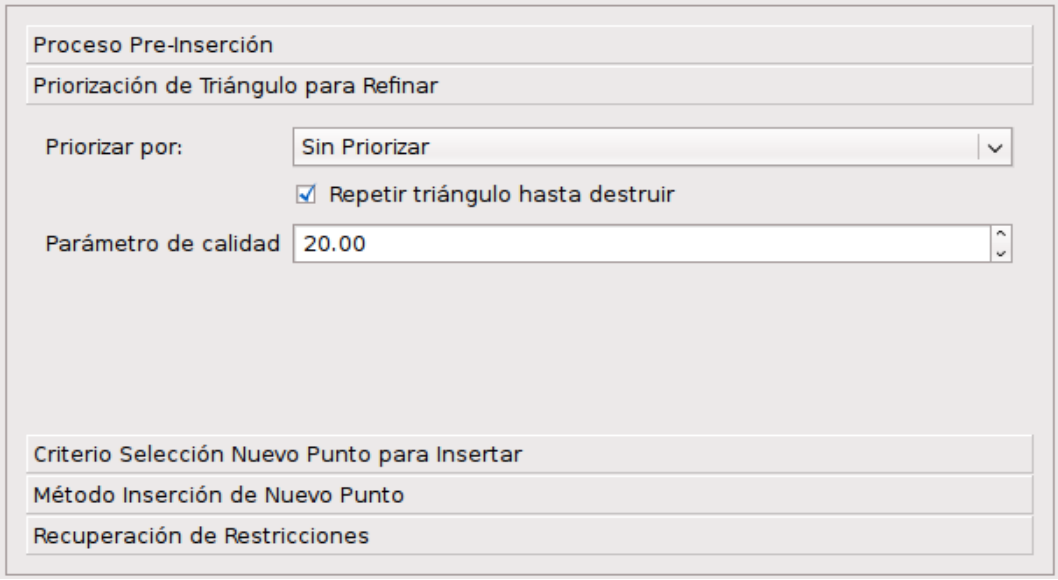


Figura 26: GUI Configuración

Separado en 5 categorías, una para cada paso del proceso de refinamiento. En la imagen se muestra la sección de algoritmos de selección de triángulos.

3.6.3. Control

En el sector de Control se puede modificar la escala de la visualización de la malla y la posición del punto central que se visualiza (coordenada en un plano). Además, se tiene la opción de realizar un control manual o automático en el refinamiento. En la Figura 27 se detallan los elementos de Control.

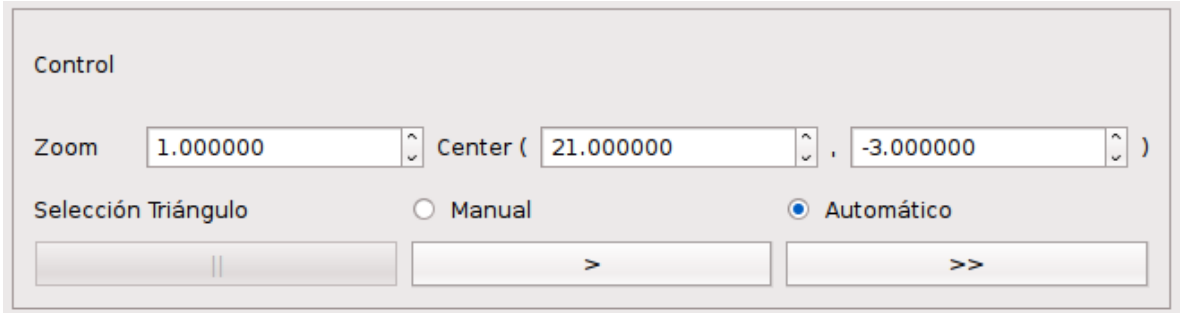


Figura 27: GUI Control

3.6.3.1. Control Manual

Al seleccionar la opción Manual en el Control, el refinamiento se llevará a cabo en el triángulo que se encuentre seleccionado en la malla con el puntero. Esta opción está pensada para analizar paso a paso el resultado de cada inserción.

3.6.3.2. Control Automático

El refinamiento automático utiliza el algoritmo de selección de triángulo especificado en la Configuración. El refinamiento puede realizarse con el botón Avanzar, o puede iterar hasta que el algoritmo de selección no encuentre algún otro triángulo de mala calidad con respecto al parámetro de la Configuración. Además, existe la opción de iterar un mismo triángulo hasta destruirlo, cómo es aplicado en los algoritmos Lepp, los cuales iteran hasta que el triángulo de mala calidad no exista en la malla.

En este punto es importante que los algoritmos de selección de triángulos para refinar, consideren la existencia de triángulos que no permitan mejorar la calidad de la malla (por ejemplo, dos bordes restringidos que generen un ángulo menor al exigido), para evitar caer en ciclos.

3.6.4. Información

En el sector información se mantiene la información actualizada de la malla después de cargar una malla y después de cada proceso de refinamiento que se realice, sea este manual o automático. La información desplegada tiene la siguiente estructura:

==========
Archivo: /home/alvaro/workspace/MeshSuite/gui/data/10.mesh
Ángulo Mínimo: 25.0028
Triángulos: 292
Vértices: 169
==========
Pre-proceso:: Ninguno
Selección Triángulo: No Priorizar
Ángulo exigido: 25°
Nuevo Punto: Lepp-Centroide-Delaunay
Tipo Inserción: Con Intercambio Diagonales
Ángulo Mínimo: 29.884
Triángulos: 14
Vértices: 12
==========

CAPÍTULO 4

EXPERIMENTACIÓN

Tanto para validar la correcta implementación de los algoritmos de refinamiento estudiados en esta memoria e implementados en el marco de experimentación, para validar el proceso de refinamiento propuesto junto a su extensibilidad y también para probar que el marco de experimentación permite estudiar el comportamiento de algún algoritmo y modificar algún elemento del proceso de refinamiento, se seleccionaron tres colecciones de vértices y segmentos de distinta procedencia y características:

- la malla A es extraída directamente desde las mallas de ejemplo del software Triangle ([She96]) y consta de 29 puntos, 29 triángulos y 1 hueco (Figura 28a)

- la malla B se generó a partir de un ejemplo usado para mostrar los resultados de la implementación del algoritmo Off-Center ([Ung04]) y consta de 72 puntos, 80 triángulos y 4 huecos (Figura 28b)

- la malla C ha sido generada con 50 puntos aleatorios y 84 triángulos (Figura 28)

Todos los refinamientos realizados a continuación fueron realizados tomando éstas mallas como mallas iniciales.

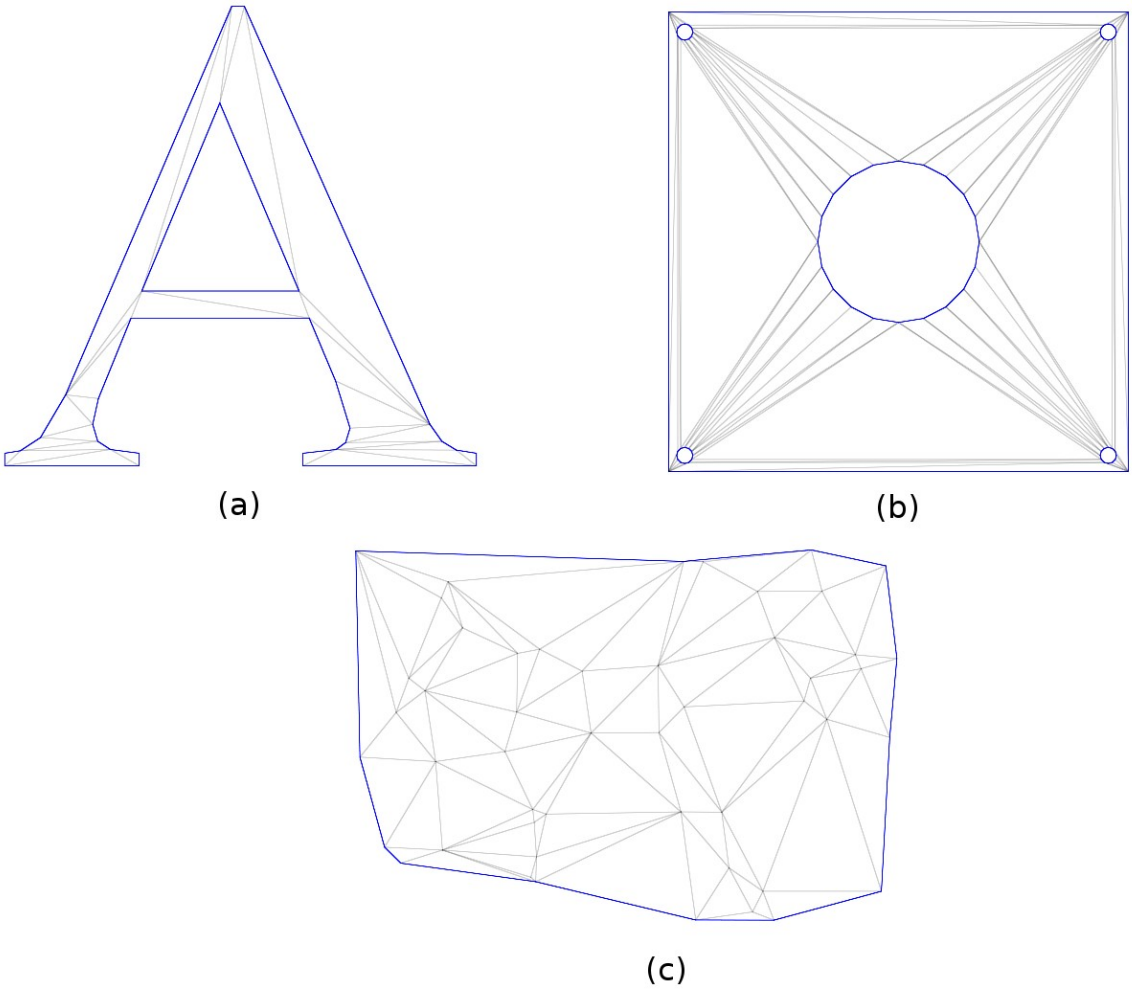


Figura 28: Mallas para validación y experimentación

4.1. Validación de algoritmos implementados

Para validar la implementación del proceso de refinamiento y de su correcto funcionar al momento de ejecutar un algoritmo, se ha procedido a comparar los resultados de la generación de mallas y refinamiento resultantes del marco de experimentación con los resultados generados por el software Triangle sobre las mismas mallas. Triangle es una herramienta de altísima calidad y

eficiencia desarrollada por Jonathan Shewchuk escrita en el lenguaje C. Esta aplicación cuenta con una gran precisión al usar operaciones de punto flotante exactas para evitar en la mayor forma posible los problemas de precisión y también utiliza predicados geométricos robustos, como el test de circuncírculo para saber si un punto yace al interior del circuncírculo de un triángulo. Los predicados geométricos documentados por Shewchuk ([She09]) son implementados en el marco de experimentación.

Triangle tiene implementado el algoritmo Off-Center y selecciona en cada iteración del refinamiento el triángulo de mala calidad con la arista de menor tamaño. La configuración del proceso de refinamiento del marco de experimentación fue el siguiente:

(i) Proceso Pre-Inserción: Arreglar aristas encroached

(ii) Criterio de selección de triángulo para refinar: Priorizar aristas de menor tamaño

(iii) Criterio de selección de nuevo punto para insertar: Off-Center - Delaunay

(iv) Método de inserción de un nuevo punto a la malla: Intercambio de diagonales

(v) Recuperación de aristas restringidas: Ninguno

Las comparaciones se adjuntan en los gráficos de las figuras 29, 30 y 31 (tablas fuentes en Apéndice A).

La comparación está realizada en base al número de vértices y al número de triángulos presentes en la malla final.

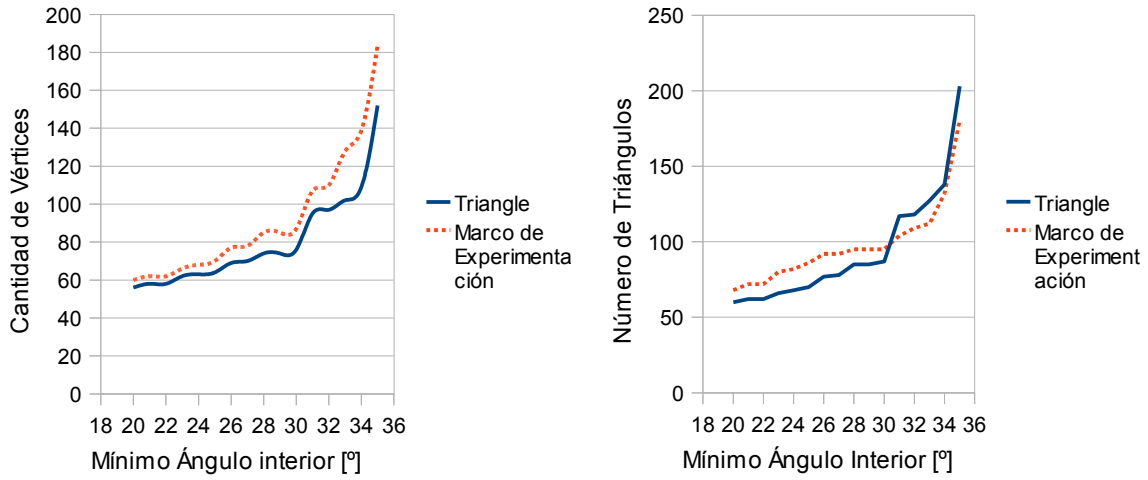


Figura 29:Triangle vs. Marco de Experimentación: Off-Center-Delaunay malla A

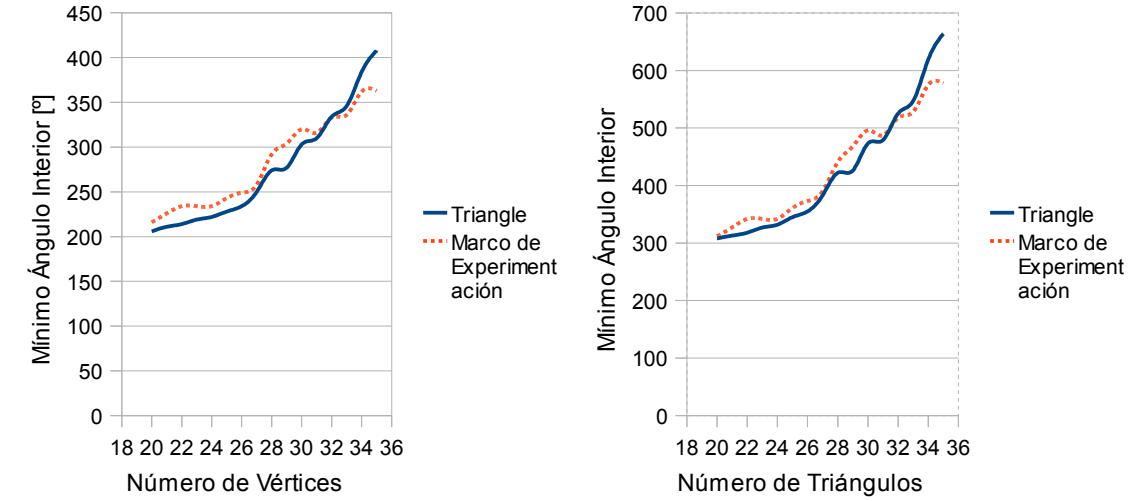


Figura 30:Triangle vs. Marco de Experimentación: Off-Center-Delaunay malla B

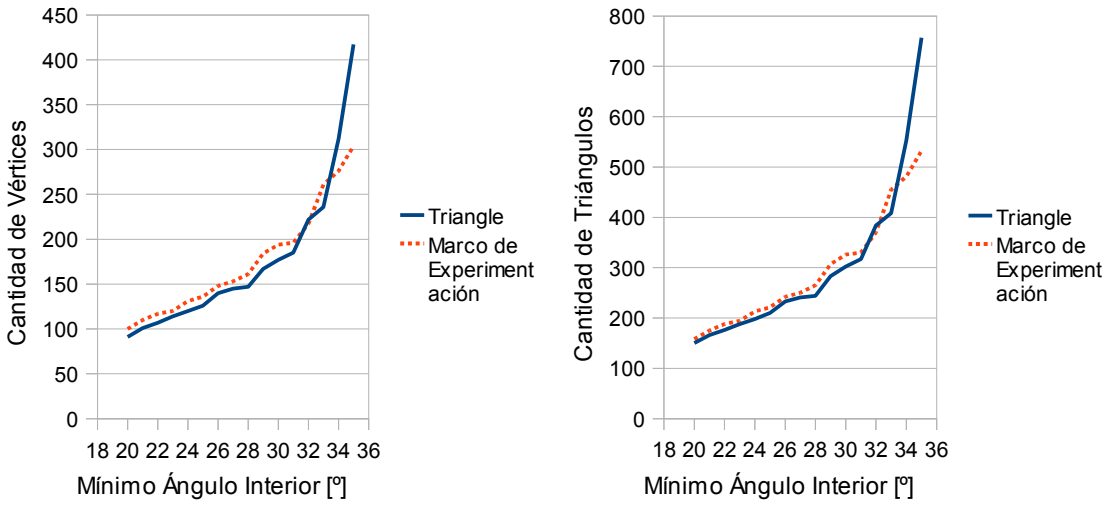


Figura 31:Triangle vs. Marco de Experimentación: Off-Center-Delaunay malla C

Para que la comparación no se basara sólo en un algoritmo, se modificó el código fuente de Triangle para que usara exclusivamente el circuncentro del triángulo para refinar en vez del Off-Center, por lo que se eliminó el uso del off-center. En este caso, se modifica el criterio de selección de nuevo punto para usar el Circuncentro-Delaunay. Las comparaciones se adjuntan en los gráficos de las figuras 32, 33 y 34 (tablas con datos de origen en Apéndice B).

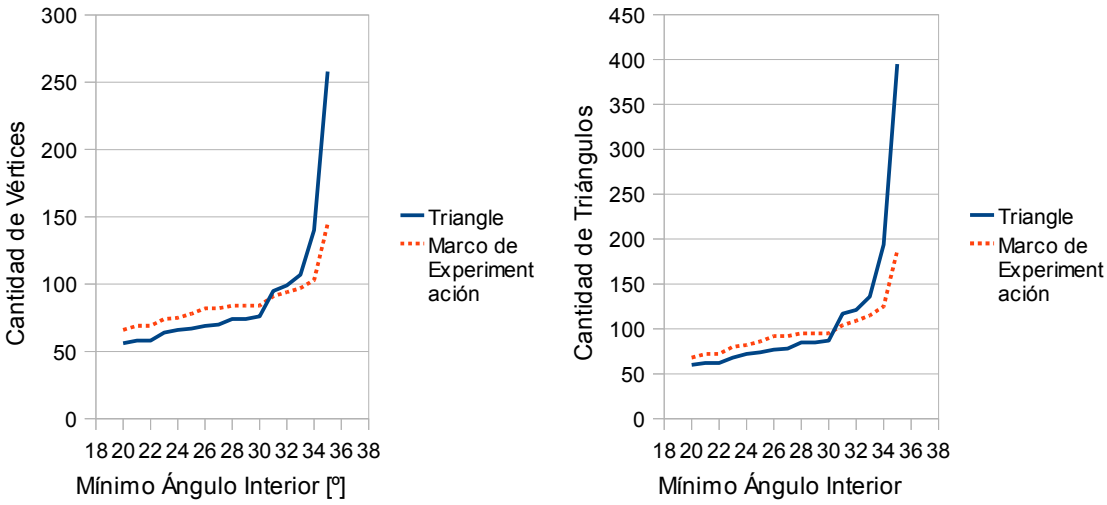


Figura 32:Triangle vs. Marco de Experimentación: Circuncentro-Delaunay malla A

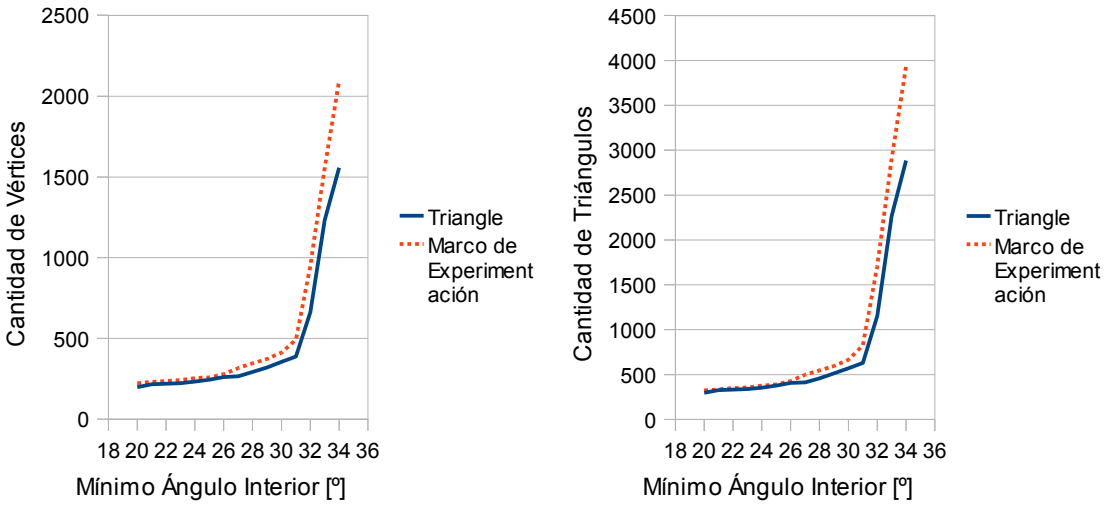


Figura 33:Triangle vs. Marco de Experimentación: Circuncentro-Delaunay malla B

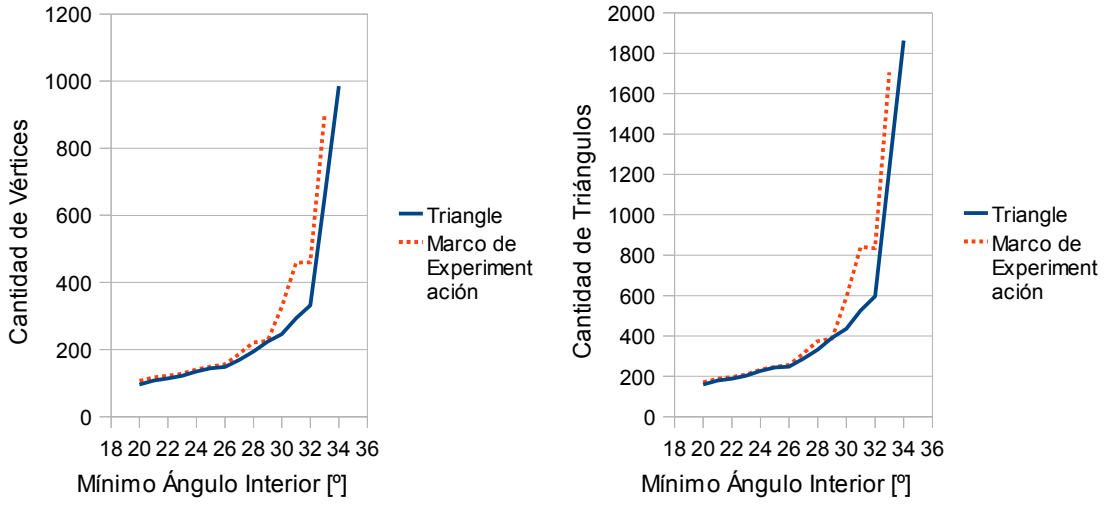


Figura 34:Triangle vs. Marco de Experimentación: Circuncentro-Delaunay malla C

Con ambos algoritmos, se puede apreciar que los resultados son comparables. Para el algoritmo Off-Center - Delaunay, el marco de experimentación comenzó generando levemente una cantidad mayor de triángulos y un mayor número de inserciones, diferencia que aproximadamente a partir

de un ángulo de exigencia de 30º se invierte en favor del marco de experimentación en las mallas estudiadas. Para el algoritmo Circuncentro-Delaunay, el comportamiento anterior sólo se cumplió con la malla B, mientas que con el resto, los resultados del marco de experimentación estuvieron levemente por sobre los resultados de Triangle. Las diferencias en los resultados de ambas herramientas, especialmente la menor cantidad de vértices y triángulos generados por Triangle, puede ser explicado por la gran madurez que tiene esta herramienta, la cual ha sido perfeccionada durante años, incluyendo optimizaciones en todo el proceso de refinamiento, especialmente en el ordenamiento y priorización de triángulos para refinar.

Con las dos aplicaciones, los valores obtenidos son los que se lograron calcular antes de que éstas arrojaran un error y/o advertencia de pérdida de precisión en los resultados, lo que usualmente ocurre cuando se intenta insertar en la malla un punto que ya existe. En el uso del algoritmo Circuncentro - Delaunay quedó en manifiesto los problemas de precisión que arrastra el marco de experimentación en desmedro del uso de aritmética exacta de Triangle, ya que el marco de experimentación perdió precisión al tratar de refinar con un ángulo de 36º en este caso con las tres mallas estudiadas.

4.2. Extensión con Nuevos Algoritmos

Con los algoritmos estudiados ya implementados y funcionales, el siguiente paso consiste en validar el diseño utilizado, extendiendo las opciones de sub-algoritmos disponibles en los pasos del proceso de refinamiento.

Producto de la implementación de los algoritmos basados en Lepp y el algoritmo de Ruppert, resultó natural mezclar las alternativas disponibles para cada paso. Se propone un algoritmo Lepp - Circuncentro - Delaunay, que consiste en:

1. Proceso pre-inserción: utilizar el proceso proveniente del algoritmo de Ruppert.

2. Selección de triángulo para refinar: prioridad a triángulos con el menor ángulo interior.

3. Selección de nuevo punto para insertar: por cada triángulo de mala calidad T, encontrar el circuncentro C del triángulo terminal del lepp asociado T. Si alguna arista restringida se convierte en una arista encroached por causa de C, seleccionar el punto medio de la arista encroached como nuevo punto para insertar. En caso contrario, seleccionar C.

4. Inserción de nuevo punto: realizar alguna inserción Delaunay.

5. Recuperación de aristas: ninguna en particular.

En los pasos 1, 2, 3 y 5 se reutilizan los procedimientos que ya se encuentran implementados. En el paso de selección del nuevo punto para insertar es necesario extender de la aplicación con este algoritmo propuesto. Esto se reduce a realizar una nueva implementación de la *interface* de selección de nuevo punto. Es más, la codificación se limita básicamente a copiar el algoritmo de

Ruppert y antes de buscar el nuevo punto, reemplazar el triángulo seleccionado por el triángulo terminal del Lepp.

Los resultados de este experimento se pueden ver en las Figuras 35 y 36 (tablas fuentes en Apéndice B), donde se puede apreciar que el algoritmo alcanza refinamientos con ángulos mínimos mayores a 37º con un número de inserciones razonable para las mallas A y B. Ninguno de los algoritmos estudiados e implementados anteriormente logró resultados con 37º sobre las mismas mallas. En la Figura 37 se muestra el resultado de refinar la malla C con una exigencia de 37º (Mallas A y B en Apéndice D).

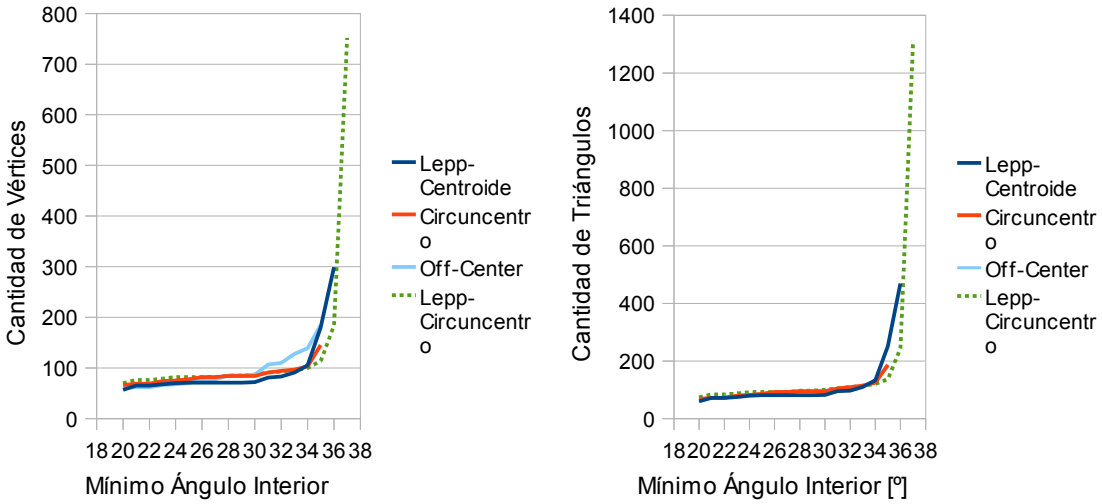


Figura 35:Rendimiento Lepp-Circuncentro Malla A

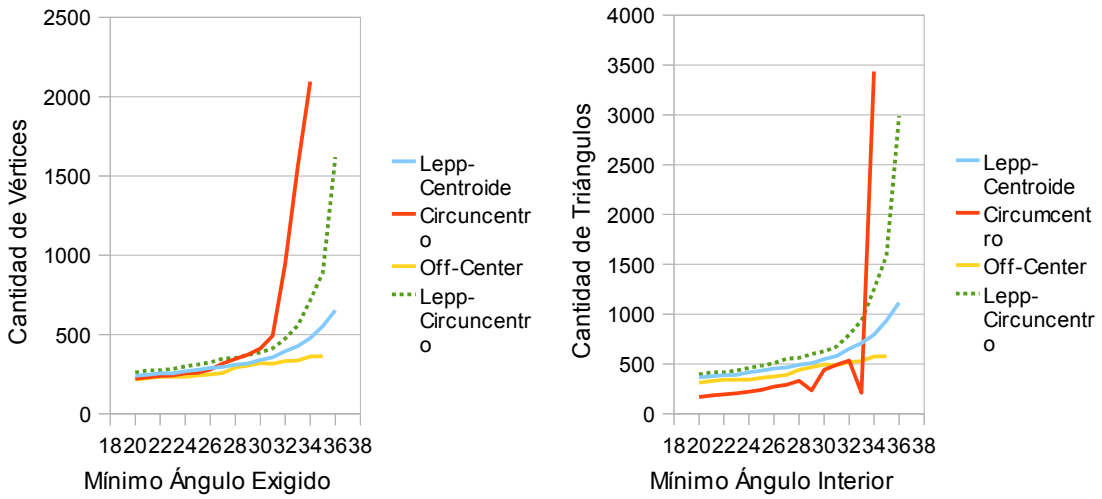


Figura 36:Rendimiento Lepp-Circuncentro Malla B

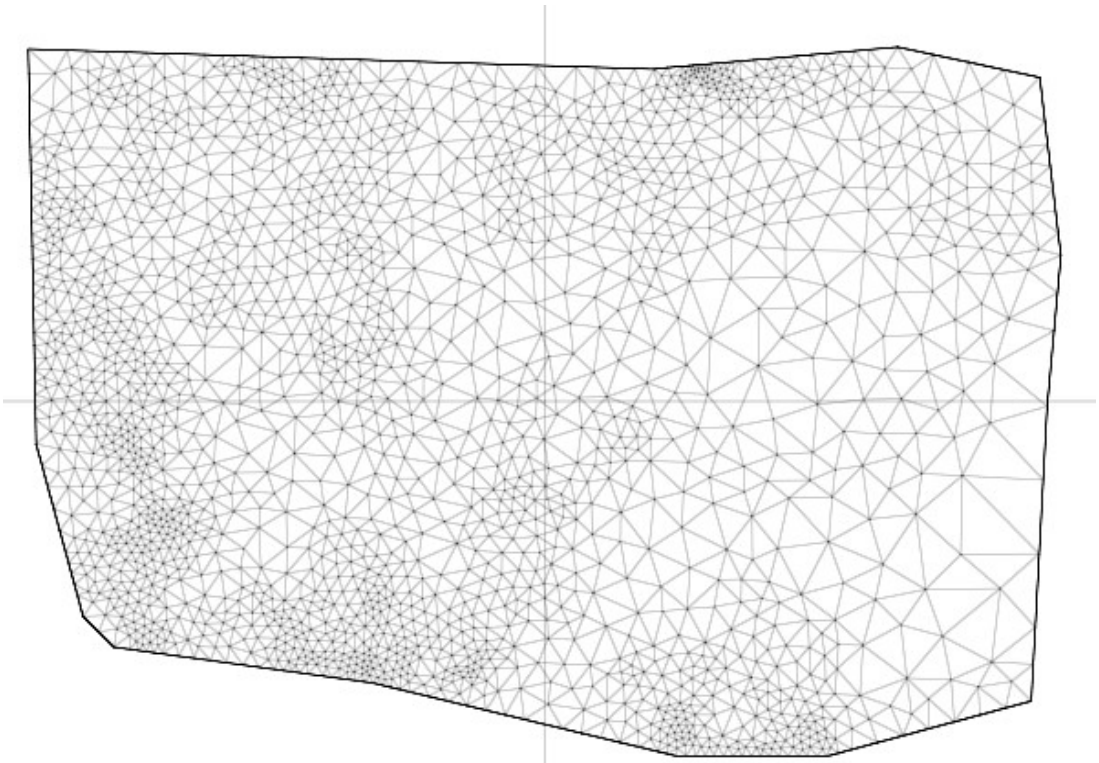


Figura 37:Resultado Lepp - Circuncentro - Delaunay en Malla C con exigencia 37º
1972 vértices y 3673 triángulos

4.3. Inserciones en los bordes

Para estudiar el comportamiento de los algoritmos en los bordes, se decidió extender el marco de experimentación con dos nuevos algoritmos de selección de triángulos. El primer algoritmo, basado en una idea comentada en [Ung04], consiste en priorizar los triángulos de mala calidad con menor circunradio. El segundo algoritmo consiste en priorizar en primer lugar los triángulos que posean algun borde de malla y luego priorizar por menor longitud de arista.

Para extender la aplicación, sólo es necesario implementar la *interface* asociada a la selección de triángulos. Por ejemplo, el algoritmo de selección de triángulos con menor circuncentro, básicamente la implementación se limita al siguiente código:

```
class TriangleSelectionSmallestCircumradius: public TriangleSelection{
public:
    TriangleSelectionSmallestCircumradius();
    Triangle* process(Mesh* mesh, double value){
            Triangle* ret = 0;
            double minCircumradius = MAX_INT;
            foreach(Triangle* t, mesh->triangles()){
                  if( t->getSmallestAngleValue() < value &&
                        t->circumradius() <= minCircumradius){
                        minCircumradius = t->circumradius();
                        ret = t;
                  }
            }
            return ret;
      }
    ~TriangleSelectionSmallestCircumradius();
}
```

Para este algoritmo, los resultados obtenidos se pueden visualizar en las figuras 38 y 39 (tablas fuentes en Apéndice C), en la cual se muestra una comparación entre algoritmos de selección considerando la cantidad de vértices y de triángulos de la malla resultante.

Se puede apreciar que el algoritmo Lepp - Centroide no se ve particularmente mejorado por

alguna priorización que se ocupe en la selección de triángulos de mala calidad, ya que todos los métodos de priorización producen salidas similares a no priorizar (inclusive peores). Por otro lado, el algoritmo Off-Center - Delaunay deja en manifiesto que la priorización sí es importante, ya que en su caso, la priorización por arista de menor longitud genera resultados notoriamente mejores.

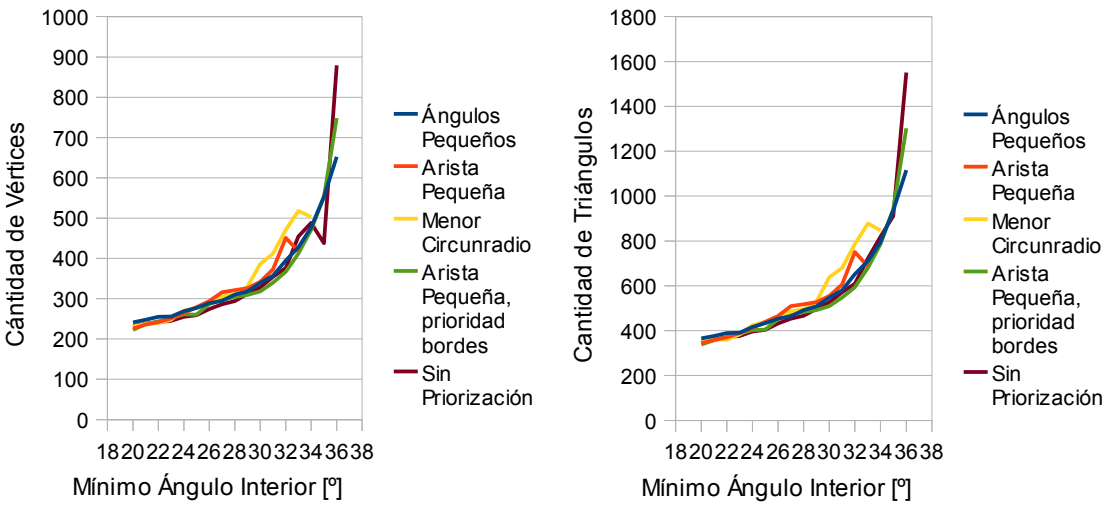


Figura 38: Comparación algoritmos selección de triángulo en Lepp - Centroide - Delaunay en Malla B

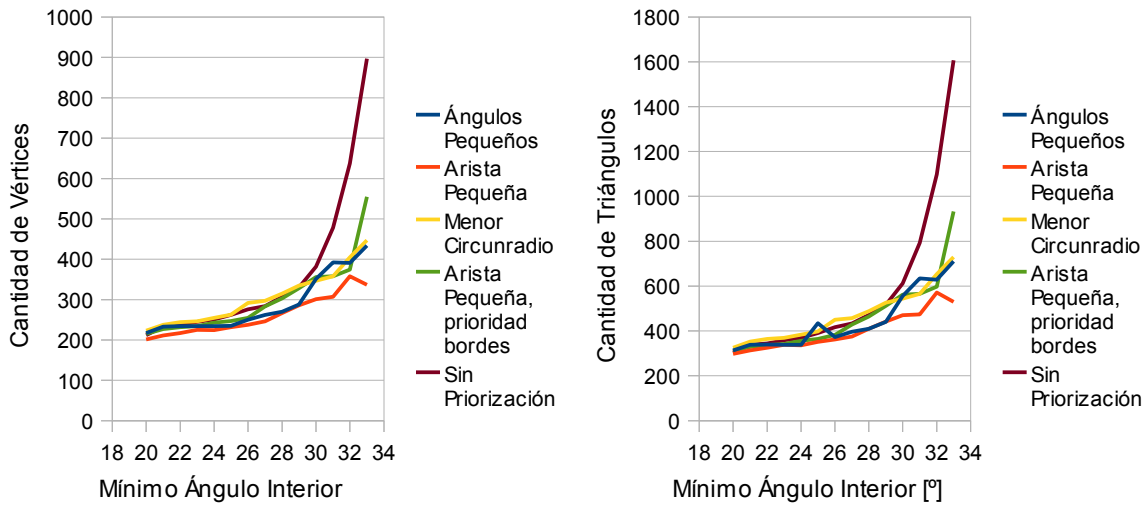


Figura 39: Comparación algoritmos selección de triángulo en Off-Center - Delaunay en Malla B

CAPÍTULO 5

DISCUSIÓN Y CONCLUSIONES

El marco de experimentación propuesto y desarrollado por esta memoria permite trabajar y experimentar con distintos algoritmos y posee un diseño que permite expansibilidad, aceptando la inclusión de nuevos algoritmos y técnicas de una manera simple y localizada. En particular, la herramienta es fácilmente extensible con:

- Nuevos procesos pre-inserción, los cuales pueden ser aplicados sólo antes de la primera inserción o antes de cualquier inserción.

- Nuevos métodos de selección de triángulos para refinar, con el fin de priorizar de distintas formas el orden en que se refinan los triángulos de mala calidad.

- Nuevos criterios para elegir un nuevo punto para insertar en la malla, tomando como referencia el triángulo entregado por el método de selección de triángulos.

- Nuevos algoritmos de inserción de puntos a una malla, sean éstos Delaunay o No Delaunay.

- Nuevos algoritmos de recuperación de aristas restringidas.

El diseño e implementación del proceso de refinamiento, basado en patrones de programación con orientación a objetos, permite combinar las distintas implementaciones para cada uno de los pasos definidos.

La estructuras de datos utilizadas son simples y proporcionan un ambiente de testeo y de implementación común para todos los algoritmos implementados.

La interfaz gráfica que se provee permite realizar un seguimiento directo sobre todo el proceso de refinamiento, además de dar control sobre el refinamiento y la inserciones que se realicen

Los algoritmos de refinamiento disponibles actualmente en la herramienta desarrollada son:

- Circuncentro - Delaunay
- Lepp - Bisección
- Lepp - Punto Medio - Delaunay
- Lepp - Centroide - Delaunay
- Off-Center - Delaunay
- Lepp - Circuncentro - Delaunay

Los algoritmos estudiados, fueron agregados al marco de experimentación de una forma simple y elegante, extendiendo la aplicación sólo en lugares específicos y acotados.

Un algoritmo resultante de la experimentación (Lepp - Circuncentro - Delaunay) obtuvo interesantes resultados logrando finalizar correctamente exigiendo un ángulo mínimo interior superior a 37º, lo que es un avance si se compara con los algoritmos que se usaron como referencia, ya que ninguno de éstos logró resultados a esa exigencia.

Se estudió un caso de inserción en el borde de la malla, agregando nuevos métodos de priorización de triángulos de mala calidad para refinar. Se ingresaron dos criterios nuevos:

- Priorizar los triángulos de mala calidad con menor circuncentro

- Priorizar los triángulos con arista de menor longitud y además dar prioridad a las aristas que sean borde de malla

Los resultados de éstas criterios no mostraron avances en ninguno de los algoritmos estudiados, en comparación a las priorizaciones previamente implementadas.

CAPÍTULO 6

TRABAJO FUTURO

Entre los aspectos que se pueden mejorar en la herramienta desarrollada, se encuentra el orden de los algoritmos de selección de triángulos para refinar, ya que en ese paso del proceso de refinamiento, al recibir una triangulación sin ningún ordenamiento, cualquier priorización que se desee implementar, será de orden lineal al número actual de triángulos presentes en la triangulación.

La usabilidad de la herramienta también tiene varios elementos que pueden ser mejorados, entre los que se incluyen:

- Carga de mallas y conjuntos de vértices y aristas iniciales, ya que actualmente sólo reciben un tipo de formato.
- Un control que permita deshacer cambios realizados a la malla.
- Herramientas para obtener estadísticas de los algoritmos o combinaciones utilizadas
- Carga y guardado de combinaciones.

APÉNDICE A

Comparación entre los valores entregados por Triangle Vs. Marco de Experimentación. Los valores ND indican que la aplicación se quedó sin precisión o que no finalizó correctamente.

comparación	Malla A	Off-Center-Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	56	60	60	68
21	58	62	62	72
22	58	62	62	72
23	62	66	66	80
24	63	68	68	82
25	64	70	70	86
26	69	77	77	92
27	70	78	78	92
28	74	85	85	95
29	74	85	85	95
30	76	87	87	95
31	95	107	117	104
32	97	110	118	109
33	102	128	127	112
34	109	139	138	132
35	152	184	203	180

comparación	Malla A	Circuncentro-Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	56	66	60	68
21	58	69	62	72
22	58	69	62	72
23	64	74	68	80
24	66	75	72	82
25	67	78	74	86
26	69	82	77	92
27	70	82	78	92
28	74	84	85	95
29	74	84	85	95
30	76	84	87	95
31	95	91	117	104
32	99	94	121	109
33	107	97	136	115
34	140	103	194	125
35	258	145	395	187
36	ND	ND	ND	ND

comparación	Malla B	Off-Center-Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	206	216	308	312
21	211	226	313	327
22	214	234	318	342
23	219	234	327	342
24	222	234	332	342
25	228	243	345	361
26	234	249	355	373
27	250	258	383	390
28	274	292	422	441
29	277	304	426	468
30	303	320	473	496
31	310	316	479	487
32	334	333	525	518
33	346	336	547	528
34	384	362	619	575
35	408	363	664	578

comparación	Malla B	Circuncenter-Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	199	223	296	324
21	216	231	326	336
22	219	238	332	350
23	224	242	340	355
24	233	255	355	378
25	245	259	378	390
26	262	279	406	430
27	266	317	412	499
28	292	346	457	546
29	321	373	513	596
30	355	412	570	665
31	388	492	630	825
32	663	947	1152	1703
33	1234	1557	2266	2902
34	1557	2095	2884	3937
35	ND	ND	ND	ND

comparación	Malla C	Off-Center – Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	91	100	150	158
21	101	110	166	175
22	107	117	176	188
23	114	120	188	194
24	120	131	198	213
25	126	136	210	221
26	140	148	233	242
27	145	153	241	250
28	147	161	244	265
29	167	184	283	307
30	177	194	302	326
31	185	196	317	330
32	222	217	384	370
33	236	261	408	455
34	312	276	553	481
35	417	304	757	532

comparación	Malla C	Circuncentro - Delaunay		
		Vértices	Triángulos	
grados	Triangle	Marco de Experimentación	Triangle	Marco de Experimentación
20	96	107	160	171
21	108	117	180	188
22	114	122	189	197
23	122	128	203	209
24	135	141	227	233
25	144	149	244	247
26	148	156	248	257
27	169	187	287	314
28	194	221	333	375
29	224	226	391	386
30	246	328	436	593
31	294	459	527	841
32	332	460	597	834
33	656	901	1226	1706
34	985	ND	1864	ND
35	ND	ND	ND	ND

APÉNDICE B

Comparación de rendimiento entre algoritmos de refinamiento. Los valores ND indican que la aplicación se quedó sin precisión o que no finalizó correctamente. Los algoritmos que Lepp Centroide y Lepp-Circuncentro no utilizaron ninguna priorización en particular. Los algoritmos Off-Center y Circuncentro-Delaunay utilizaron priorización de aristas de menor tamaño.

comparación	Malla B							
	Vértices				Triángulos			
grados	Lepp-Centroide	Circumcenter	Off-Center	Lepp-Circuncentro	Lepp-Centroide	Circumcenter	Off-Center	Lepp-Circuncentro
20	241	223	216	261	366	168	312	394
21	248	231	226	274	376	183	327	415
22	255	238	234	275	389	196	342	417
23	256	242	234	284	391	205	342	434
24	269	255	234	299	417	221	342	461
25	277	259	243	311	433	239	361	483
26	289	279	249	325	454	272	373	509
27	295	317	258	348	465	290	390	553
28	310	346	292	352	492	331	441	561
29	318	373	304	372	508	234	468	599
30	339	412	320	387	548	440	496	626
31	357	492	316	413	580	491	487	674
32	395	947	333	475	652	536	518	792
33	427	1557	336	557	710	210	528	938
34	477	2095	362	718	795	3437	575	1252
35	553	ND	363	890	936	ND	578	1591
36	652	ND	ND	1619	1116	ND	ND	2990
37	ND	ND	ND	ND	ND	ND	ND	ND

comparación	Malla A							
	Vértices				Triángulos			
grados	Lepp-Centroide	Circumcenter	Off-Center	Lepp-Circuncentro	Lepp-Centroide	Circumcenter	Off-Center	Lepp-Circuncentro
20	57	66	60	70	60	68	68	74
21	65	69	62	76	72	72	72	84
22	65	69	62	76	72	72	72	84
23	68	74	66	79	75	80	80	87
24	70	75	68	82	79	82	82	92
25	71	78	70	82	81	86	86	92
26	71	82	77	82	81	92	92	92
27	71	82	78	82	81	92	92	92
28	71	84	85	85	81	95	95	97
29	71	84	85	85	81	95	95	97
30	72	84	87	87	82	95	95	100
31	81	91	107	91	95	104	104	106
32	83	94	110	93	97	109	109	109
33	91	97	128	96	110	115	112	114
34	105	103	139	100	133	125	132	120
35	180	145	184	114	252	187	180	137
36	299	ND	ND	184	469	ND	ND	245
37	ND	ND	ND	752	ND	ND	ND	1313

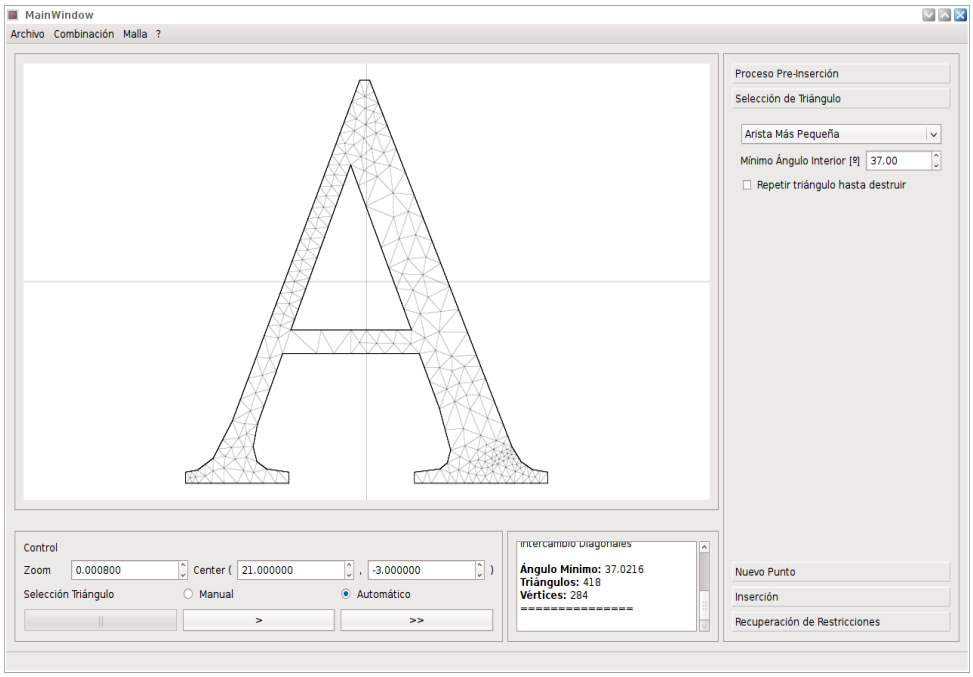
APÉNDICE C

Comparaciones de rendimiento de Lepp – Centroide -Delaunay usando distinta priorización de selección de triángulo para refinar. Los valores ND indican que la aplicación se quedó sin precisión o que no finalizó correctamente.

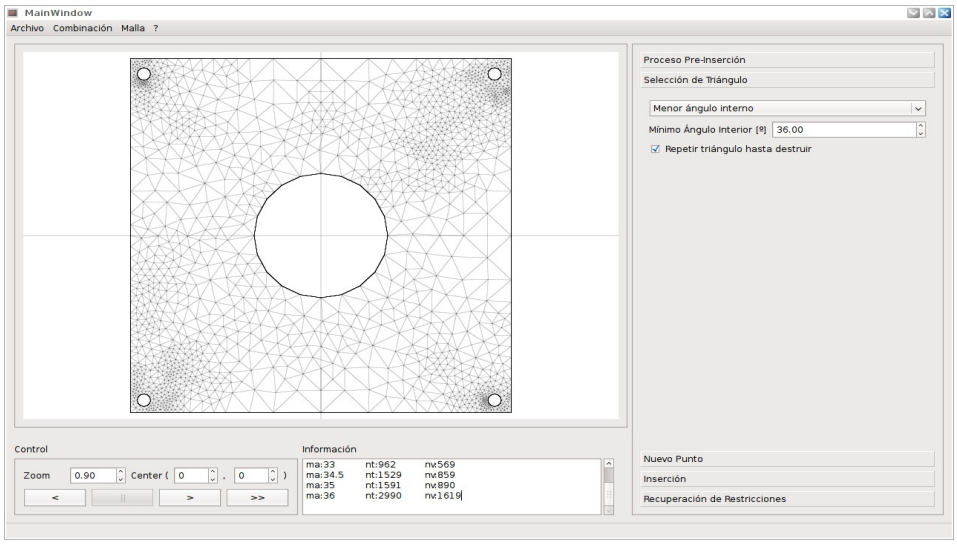
Lepp-Centroide	Malla B									
	vertices					triángulos				
grado	Ángulos Pequeños	Arista Pequeña	Menor Circunradio	Arista Pequeña, Bordes	Sin Priorización	Ángulos Pequeños	Arista Pequeña	Menor Circunradio	Arista Pequeña, Bordes	Sin Priorización
20	241	226	231	223	232	366	344	353	338	352
21	248	236	237	237	238	376	359	361	357	360
22	255	242	240	244	242	389	371	360	370	370
23	256	251	248	252	245	391	389	384	386	376
24	269	264	270	261	255	417	412	424	403	396
25	277	279	278	260	259	433	440	440	405	404
26	289	294	291	287	274	454	466	462	451	432
27	295	316	306	297	286	465	509	486	471	453
28	310	321	311	304	294	492	518	495	483	467
29	318	326	330	309	312	508	527	530	492	498
30	339	342	386	318	328	548	555	638	509	526
31	357	373	412	340	354	580	606	680	547	571
32	395	451	471	367	376	652	751	786	594	609
33	427	418	518	412	454	710	687	878	681	722
34	477	ND	503	471	488	795	ND	847	785	818
35	553	ND	ND	553	438	936	ND	ND	942	911
36	652	ND	ND	748	879	1116	ND	ND	1303	1551

Off-Center Delaunay	Malla B									
	vertices					triángulos				
grado	Ángulos Pequeños	Arista Pequeña	Menor Circunradio	Arista Pequeña, Bordes	Sin Priorización	Ángulos Pequeños	Arista Pequeña	Menor Circunradio	Arista Pequeña, Bordes	Sin Priorización
20	217	201	223	215	212	313	298	325	311	304
21	233	211	238	227	233	339	314	353	329	339
22	234	217	244	232	237	340	325	365	337	345
23	234	225	246	234	245	338	339	369	342	359
24	234	224	255	243	251	338	336	384	356	369
25	235	232	263	247	262	435	351	398	364	391
26	250	238	292	255	276	373	362	450	382	417
27	262	246	297	283	284	396	375	458	429	434
28	270	267	315	303	307	410	410	488	465	472
29	288	286	335	328	330	441	442	526	512	517
30	351	301	347	356	382	556	470	544	562	612
31	392	307	358	358	478	634	474	566	566	793
32	391	358	405	374	637	629	571	654	598	1098
33	434	337	447	555	897	711	530	730	933	1607
34	1077	374	543	885	ND	1967	600	909	1557	ND
35	ND	385	ND	ND	ND	ND	621	ND	ND	ND

APÉNDICE D



Malla A refinada a 37º con Lepp – Circuncentro – Delaunay



Malla B refinada a 36º con Lepp – Circuncentro – Delaunay

REFERENCIAS BIBLIOGRÁFICAS

[Law72] Charles L. Lawson: Transforming triangulations. Discrete Mathematics, 3(4), 365–372. 1972.

[Wat81] D.F. Watson: Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes. The Computer Journal, 24(2), 167-172. 1981.

[Rup95] Ruppert J.: A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. JOURNAL OF ALGORITHMS, 18(3), 548. 1995.

[Azo10] David Azocar, Marcelo Elgueta, María Cecilia Rivara: Automatic LEFM crack propagation method based on local Lepp-Delaunay mesh refinement. Advances in Engineering Software, 41(2), 111-119. 2010.

[Riv97] Rivara M-C: New Longest-Edge Algorithms For the Refinement and/or Improvement of Unstructured Triangulations. INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, 40(18), 3313-3324. 1997.

[Riv10] Maria-Cecilia Rivara, Carlo Calderon: Lepp terminal centroid method for quality triangulation. Advances in Geometric Modelling and Processing, 42(1), 58-66. 2010.

[Ung04] Ungor, A.: Off-Centers: A New Type of Steiner Points for Computing Size-Optimal Quality-Guaranteed Delaunay Triangulations. LECTURE NOTES IN COMPUTER SCIENCE, (2976), 152-161. 2004.

[Riv08] María-Cecilia Rivara: Lepp-bisection algorithms, applications and mathematical properties. Applied Numerical Mathematics, 59(9), 2218-2235. 2009.

[Vig97] Marc Vigo Anglada: An improved incremental algorithm for constructing restricted delaunay triangulations. COMPUTERS AND GRAPHICS, 21(1), 215-224. 1997.

[She96] Shewchuk, J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. LECTURE NOTES IN COMPUTER SCIENCE, (1148), 203-222. 1996.

[She09] Shewchuk J. R.: Lecture Notes On Geometric Predicates. 2009. Disponible en Septiembre, 2010. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.155.3956&rep=rep1&type=pdf