Universidad del Bío-Bío, Chile

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Implementación de un applet en JAVA para la visualización de algoritmos de refinamiento de mallas 2D

Proyecto presentado por Christopher Gutiérrez Mellado
para obtener el título de Ingeniero de Ejecución en Computación e Informática
Dirigida por Dr. Pedro Angel Rodríguez Moreno

2018

Agradecimientos

En primer lugar le agradezco a la vida, que me ha entregado muchas experiencias en mi corto tiempo en este lugar.

Agradezco ante todo a mi familia, mi madre Maria que siempre nos alentó a estudiar, a mi padre Jorge y mis hermanos Angélica y Alejandro.

Debo reconocimiento a los profesores que me enseñaron de diversas formas, Nelly Gómez, Sergio Bravo, Patricio Galdames, Patricio Gálvez, Oscar Gericke, Roberto Mercado, Juan Carlos Parra, Clemente Rubio, Brunny Troncoso, Marcos Iturra, Christian Vidal, Ricardo Pavéz, Ivonne Anfossi, Luis Vergara Riquelme, Lucy Concha, Gonzalo Rojas, Jazna Meza, Marcelo Pinto, Valeria Beratto, Carlos Molina, Gloria Toro, Ariel Yévenes, agradezco el haber tenido buenos profesores de los cuales recibí conocimiento y también formación personal y profesional, a mi profesor guía Pedro Angel Rodríguez Moreno, muchas gracias por su apoyo, comprensión y tiempo.

Finalmente agradezco a mis compañeros y amigos en este periodo de la vida, Álvaro Gonzales, Francisco Javier, Diego Gutiérrez, entre otros, y en especial agradezco a quien me ayudó, motivó y alentó en los momentos en que me encontraba atascado en el desarrollo del proyecto de título, Nicol Bascuñán.

Agradezco a todos ustedes por haber llegado a mi vida, no los olvidaré.

Este proyecto de título fue parcialmente financiado por el grupo de investigación Computación Ubicua, código GI 150115/EF.

Índice General

Lista de Figuras

Índice de Tablas

Índice de Clases

Índice de algoritmos

Capítulo 1

Introducción

1.1. Aspectos generales

Una malla geométrica corresponde a la discretización de un dominio geométrico Ω mediante un conjunto de elementos o formas geométricas simples (simplicies)(Ver figura A.1), tales como triángulos o cuadriláteros en dos dimensiones y tetraedros o hexaedros en tres dimensiones, donde la unica intersección de estos elementos es en un punto, arista o una cara poligonal.

Las mallas tienen asociadas un conjunto de elementos topológicos tales como: vértices, aristas, triángulos etc.

En la actualidad, las mallas se encuentran en una alta gama de aplicaciones y disciplinas, tales como simulaciones de ingeniería, aplicaciones médicas, diseño asistido por computador (CAD), métodos de elementos finitos para la interpolación de superficies y visualizaciones científicas, entre otras aplicaciones.

En su amplia gama de aplicaciones, el refinamiento de mallas geométricas se ha convertido en un objeto de estudio, el cual tiene en común el uso de los métodos de elementos finitos, éstas son técnicas numéricas para encontrar soluciones aproximadas de problemas físicos complejos modelados por ecuaciones diferenciales parciales, en los cuales se discretiza el dominio físico en subdominios más pequeños.

En general el proceso de refinamiento de mallas puede ser descrito de la siguiente forma:

1. Definir el dominio inicial en un conjunto de puntos en el plano euclidiano.
2. Estructurar el dominio para la creación de una malla inicial de buena calidad.
3. Formulación de una solución numérica desde el dominio discretizado.
4. Refinamiento de la malla y salto al paso tres para mejorar la solución.
5. Visualización de la malla y presentación de la solución.

En este proyecto de título se utilizan los algoritmos de refinamiento de mallas basados en los conceptos de LEEP (Longe Edge Propagating Path) y arista más larga, desarrollados por la profesora *Maria Cecilia Rivara* [8, 10–14].

En los algoritmos usados se incorporan dos criterios de selección de triángulos para mejorar la discretización del dominio, es decir la selección de un subdominio restringido por dos criterios, el ángulo menor de un triángulo y la distancia de la arista más larga de un triángulo. Un caso particular es la búsqueda, en una malla, de todos los triángulos que tengan, por ejemplo, en su ángulo menor, un ángulo inferior a 30 grados, para que sean refinados. Otro caso es el de la búsqueda, también en una malla, de todos los triángulos cuya arista más larga sea mayor a 20 pixeles, para que sean refinados.

En este proyecto se aportó con dos algoritmos de búsqueda, los cuales son una precondición antes de aplicar los algoritmos de refinamiento de mallas, estos algoritmos son los siguientes:

1. Distancia LongestEdge Mayores

2. Ángulo Menor De Un Triangulo

Además de generar una aplicación genérica basada en el patrón de diseño modelo-vista-controlador (MVC), con un motor gráfico generado en JAVA, el cual cuenta con los frames per second (fps) y actualización por segundo (aps).

1.2. Definición del problema

Actualmente se cuenta con un software local desarrollado en C++ por el profesor Pedro Rodríguez para el refinamiento de mallas 2D, este proyecto nace de la necesidad de migrar este software a una plataforma web por medio de un JAVA-applet, para así facilitar la accesibilidad de un software de refinamiento de mallas 2D, además de incorporar mejoras en los algoritmos de refinamiento y mejoramiento de mallas, su funcionalidad y facilidad de uso.

1.3. Objetivos

Los objetivos de este proyecto de título se han dividido de la siguiente forma.

1.3.1. Objetivo general

Migrar una aplicación C++ de refinamiento de mallas de triángulos a lenguaje JAVA, incorporando también mejoras en los algoritmos de refinamiento y mejoramiento de mallas 2D.

1.3.2. Objetivos específicos

1. Investigar lenguajes C++ y JAVA (applet) para la migración de la aplicación de mallas 2D.
2. Investigar algoritmos refinamiento de mallas de triángulos para incorporar mejoras en la aplicación final.
3. Generar un diagrama de la aplicación utilizando Lenguaje Unificado de Modelado (UML).
4. Crear un diseño de interfaz que permita la visualización de mallas 2D.
5. Obtener una aplicación genérica (applet en JAVA) que apoye la futura actualización e incorporación de nuevos algoritmos de refinamiento de mallas 2D.
6. Incorporar nuevos criterios de inserción de puntos para lograr mejoras en los algoritmos de refinamiento de mallas 2D propuestos.

1.4. Diseño de software

Patrón de diseño Modelo Vista Controlador (MVC): En líneas generales, el MVC, es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario, este patrón de diseño se utiliza para obtener una aplicación genérica con un ciclo de vida más adecuado al futuro mante-

nimiento, reutilización de código y la separación de conceptos, puesto que se separa el código en tres capas diferentes, acotadas a su responsabilidad. La capa *modelo* es la encargada de gestionar las estructuras de datos del software, la capa *vista* es la encargada de las interfaces de usuario y la capa *controlador* es la encargada de gestionar los eventos de interacción entre el usuario y el software.

1.5. Organización

Estructura de datos topológicas: En este capítulo se definen las estructuras de datos utilizadas en la creación de mallas de triángulos, también se definen las formas en que se clasifican las mallas de triángulos, mallas estructuradas y no estructuradas, también se define la creación de las mallas de elementos finitos y las mallas de diferencias finitas, para terminar con la definición del archivo de modelado en dos dimensiones, que usa el formato m2d.

Diseño del software: En el capítulo dos se explica el patrón de diseño utilizado en la creación del software, la metodología del patrón de diseño y se define la usabilidad del software mediante casos de usos.

Refinamiento de triangulaciones: En el capítulo tres se explica qué es un refinamiento de triangulaciones o mallas, y se define en qué consiste la bisección de un triángulo, también se define y explica los algoritmos utilizados para del refinamiento de mallas, LEEP-Bisección, LEEP-Centroide y LEEP-Delaunay.

Pruebas de Funcionalidad: En el capítulo cuatro se explica la funcionalidad de los algoritmos de refinamiento de mallas y las interacciones entre el usuario y el applet.

Conclusión: En el capítulo de conclusión se resume los resultados obtenidos y se muestra las principales deducciones alcanzadas de los resultados.

Capítulo 2

Estructura de datos topológicas

2.1. Estructura de datos a utilizar

La estructura de datos a usadas en el sistema desarrollado, se compone de una clase modelo. Esta clase cumple el rol de agrupar las distintas estructuras de datos y así facilitar la interacción en el patrón de diseño (MVC) (Ver Figuras 3.1,3.18), la clase modelo se compone de vértices, aristas y triángulos, los cuales sirven para representar un conjunto de polígonos irregulares en el espacio euclidiano (Ver Clase 2.1).

```
1 public class Modelo  {
2     static ArrayList<Vertice> vertices;
3     static ArrayList<Triangulo> triangulos;
4     static ArrayList<Arista> aristas;
5 };
```

Clase 2.1: Estructura del modelo a utilizar

2.1.1. Vértice

El concepto de vértice deriva del vocablo latino vertex. En la geometría, un vértice corresponde al punto de unión entre dos o mas líneas o semirrectas que originan un ángulo, la condición que debe cumplir para que se origine un vértice es que las dos rectas no sean paralelas entre sí [16] (Ver Figura 2.1).

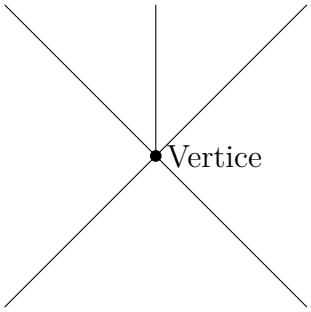


Figura 2.1: Representación de un vértice [3]

En nuestro software se utiliza la clase Vértice para representar los puntos (x,y) del espacio euclidiano, los cuales simbolizan la unión de dos o más aristas. Cada vértice se compone de una coordenada X y una coordenada Y, los cuales son de tipo Double, además cada vértice consta de un identificador de tipo entero (Ver Algoritmo 2.2).

```
1 public class Vertice {
2     private Double x;
3     private Double y;
4     private int id;
5 };
```

Clase 2.2: Clase que define un vértice

2.1.2. Arista

En geometría es el segmento de línea resultante de la intersección de dos superficies o planos, considerada por la parte exterior del ángulo que forman. Una arista también es un segmento de recta que define los lados de una figura geométrica plana [16].

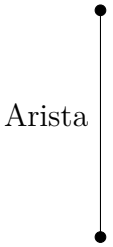


Figura 2.2: Representación de una arista [3]

Una arista tiene una longitud finita y en sus extremos está limitada por dos vértices. En nuestro software utilizaremos la clase Arista para representar las conexiones entre vértices de un polígono en el espacio euclidiano, cada arista será representada por sus dos vértices extremos de inicio y término (Ver Figura 2.2). Además, cada arista consta de un identificador de tipo entero (Ver Clase 2.3).

```
1 public class Arista {
2     private Vertice inicio;
3     private Vertice termino;
4     private int id;
5 };
```

Clase 2.3: Clase que define una arista

2.1.3. Triángulo

El concepto de triángulo deriva del vocablo latino triangulus, un triángulo es un polígono convexo[1] compuesto por tres segmentos de recta que se denominan aristas o por tres puntos en el espacio euclidiano llamados vértices [16].

Los triángulos se clasifican por sus lados o según sus ángulos. Además cada polígono de tres ángulos contiene una base, la cual corresponde a uno de sus lados o aristas. El segmento perpendicular a la base que va desde el vértice opuesto a ésta se denomina altura h de un triángulo (Ver Figura 2.3).

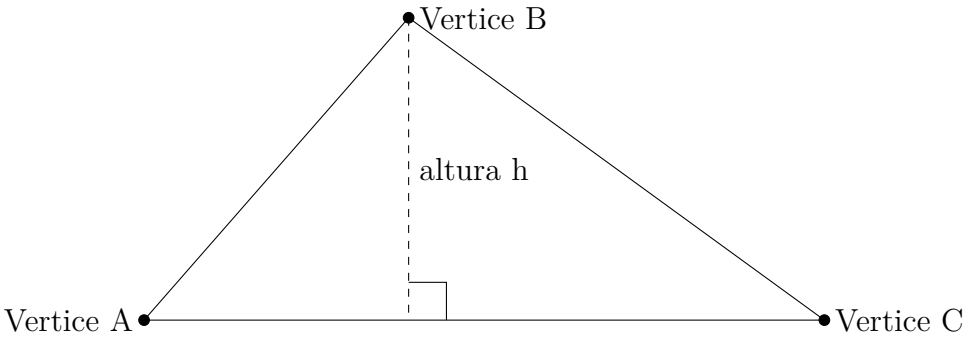


Figura 2.3: Representación de un triángulo (ABC) [3]

[1]Un polígono es estrictamente convexo si todos sus ángulos internos son estrictamente menores de 180 grados y todas sus diagonales son interiores

En nuestro software se utiliza la clase Triangulo para representar la unión de tres vértices del espacio euclidiano, cada triángulo se compone de tres Vértices y tres aristas, más tres punteros a sus triángulos vecinos, un triángulo es vecino de otro si éste comparte una arista o dos vértices con éste. Además cada triángulo consta de un identificador de tipo entero (Ver Clase 2.4).

```
1 public class Triangulo {
2     private Vertice[] vertices = new Vertice[3];;
3     private Triangulo[] vecinos = new Triangulo[3];
4     private int id;
5 };
```

Clase 2.4: Clase que define un triángulo

2.2. Representación de una triangulación o malla

La triangulación de un polígono o área poligonal representa un conjunto de n-vértices, los cuales se unen mediante aristas que no se cruzan, éstas generan un conjunto de triángulos que en su suma total de áreas es igual al área total del polígono.

Se define una triangularización a la división de una área poligonal en un conjunto de triángulos que cumplen las siguientes condiciones (Ver Figura [2.4 - 2.6]):

- La unión de todos los triángulos es igual al área del polígono original [5].

$$\mathrm{Dom} f_{(Area_{Total})} = \sum_{i=0}^{n} A(t_i)$$

donde $A(t_i) = \frac{L_i * h_i}{2}$

L_i base del triángulo t_i

h_i su altura

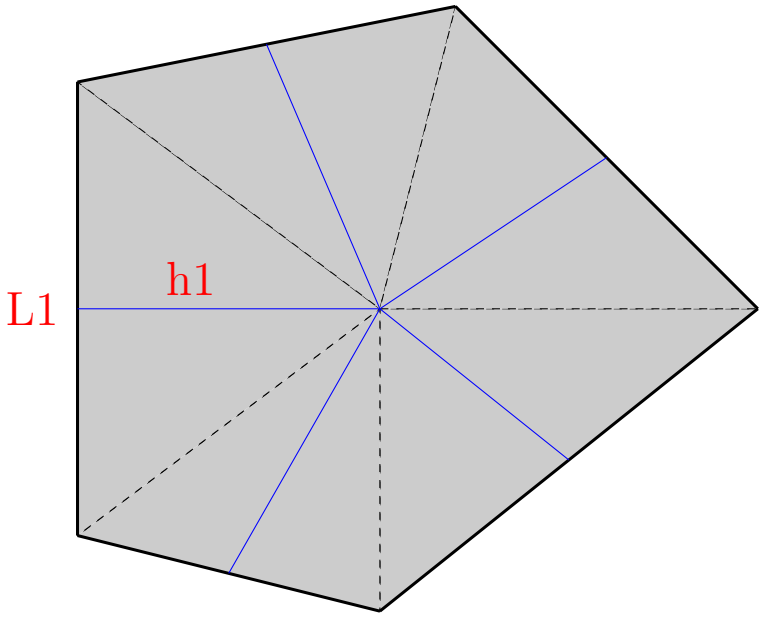


Figura 2.4: Área de un pentágono irregular[2]

[2] http://www.universoformulas.com/matematicas/geometria/triangulo/

- Los vértices de los triángulos son vértices del polígono original [5].

$$\forall(a_i, b_i, c_i) \text{ de un } t_i \; \exists(a_i, b_i, c_i) \in \mathbb{R}$$

donde

$\mathbb{R} = \sum_{i=0}^{n} A(t_i)$

$A(t_i) = \frac{1}{2}(L_i h_i)$

a_i, b_i, c_i Vértices del triángulo t_i

L_i base del triángulo t_i

h_i su altura

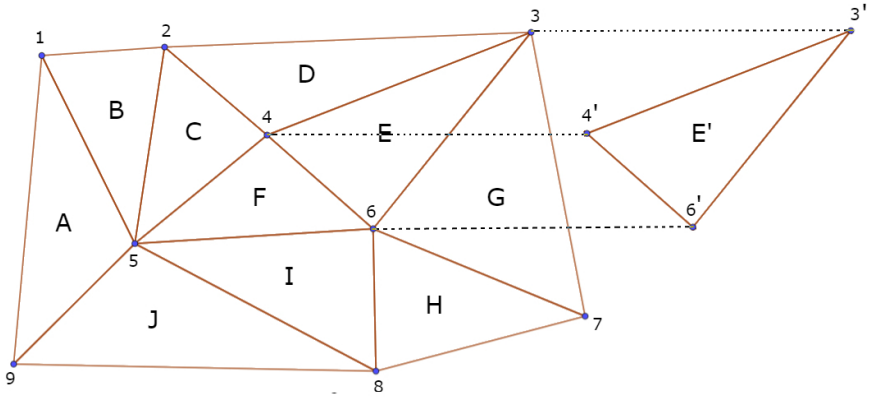


Figura 2.5: Los vértices 3',4',6', del triángulo E', corresponden a los vértices 3,4,5 del polígono original

• Cualquier pareja de triángulos es disjunta o comparte únicamente un vértice o una arista [5].

$$[(\triangle_A \cap \triangle_B) = \varnothing \mid A_i \cap A_j = \varnothing, \forall i, j \in A, i \neq j]$$
$$\wedge\, [(\triangle_A \cap \triangle_B) = Nodo \mid A_i \cap B_j = Nodo, \exists! i \in A, \exists! j \in B, i = j]$$

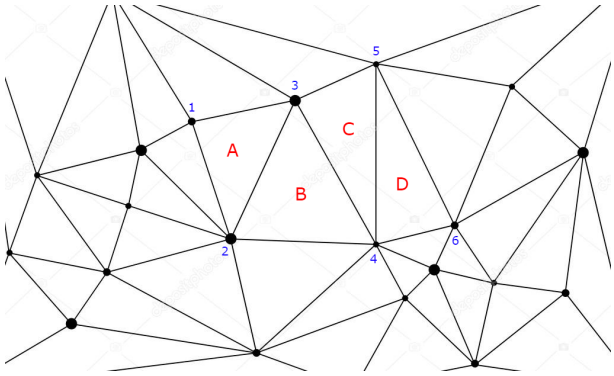


Figura 2.6: El triángulo A con vértices 1,2,3 es disjunto con el triángulo D de vértices 4,5,6 puesto que no comparten ningún vértice. Por otro lado el triángulo A de vértices 1,2,3 solo comparte un vértice con el triángulo C de vértices 3,4,5.

2.3. Clasificación de las mallas

Las mallas se clasifican en dos tipos de acuerdo a su estructura: mallas estructuradas, y mallas no estructuradas. La diferencia entre éstas radica en la forma de agrupar las celdas que conforman la malla. Éstas a su vez se subdividen por los siguientes métodos empleados en su creación.

2.3.1. Malla estructurada

Una malla estructurada es un conjunto de celdas del mismo tipo y tamaño, generalmente de cuadriláteros (2D) y hexaedros (3D) los cuales forman parte de un dominio de n-vértices conectados, usualmente la solución matemática de una malla estructurada es mediante el método de diferencias finitas. Una malla estructurada se define como la creación de una retícula que impone fuertes condiciones sobre el contorno del dominio, por lo que en muchos casos este tipo de discretización no es realizable o, siéndolo, presenta una baja calidad [9].

La solución del método de diferencias finitas se obtiene generalmente usando una malla estructurada, tal como una grilla. Éstos se pueden representar en una matriz (2D), un arreglo bidimensional (i, j) para representar los vértices de un área poligonal, donde el índice i puede utilizarse para representar los puntos en una dirección del espacio euclidiano y el índice j representa la posición de los puntos en otra dirección (Ver Figura 2.7).

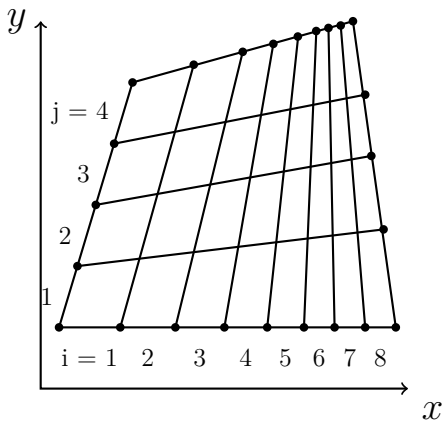


Figura 2.7: Malla cuadrilatera estructurada [17].

2.3.2. Malla no estructurada

Una malla no estructurada es un conjunto de celdas del mismo tipo pero de tamaños diferentes. En general se representan en celdas de triángulos o cuadriláteros (2D) y tetraedros o hexaedros (3D) los cuales forman parte de un dominio de n-vértices conectados. En una malla no estructurada cada celda tiene diferentes números de elementos vecinos. La solución del método de elementos finitos se obtiene generalmente usando una malla no estructurada. Este tipo de malla ofrece mucha flexibilidad para ajustarse al dominio con geometrías complejas, los elementos finitos tienen una inherente habilidad para tratar con mallas no estructuradas, estas permiten fácilmente refinamientos locales para dar resoluciones altas en regiones de interés, sin perder precisión.

En ellas se logra, por ejemplo una adecuada representación de líneas costeras en geometrías irregulares, lo cual sin duda es una ventaja frente al uso de mallas estructuradas. Crear mallas no estructuradas requiere la selección de los puntos internos de la malla (vértices) y su triangulación. La resolución debe ser tal que permita a ésta tener pequeños elementos para resolver los detalles finos de la geometría e igualmente debe poseer elementos grandes en el área del dominio donde sea posible, para reducir el número de nodos.

Para la solución numérica es necesario que los elementos de la malla tengan una alta calidad a fin de asegurar una adecuada discretización y buenas condiciones en el sistema lineal (Ver Figura 2.8).

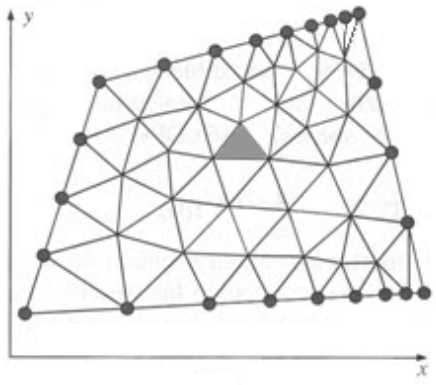


Figura 2.8: Malla triangular no estructurada [7]

2.3.3. Malla de elementos finitos

Es una malla creada por algún programa computacional, utilizada para resolver un sistema de ecuaciones diferenciales parciales (EDP) sobre un dominio dado.
Las mallas de elementos finitos son generalmente mallas no estructuradas, en la cual se crea una malla de elementos finitos para ser usada en el método de elementos finitos (MEF). Se propuso por primera vez en los años cuarenta y fue puesto en práctica en la década siguiente en el diseño aeronáutico. Desde entonces el método se ha desarrollado y aplicado extensamente en muchos otros campos, entre ellos las mallas de triángulos. Actualmente se considera un método general aplicable a la mayoría de los problemas matemáticos y de ingeniería.

El método está basado en la división del dominio continuo, sobre el que se quiere estudiar el problema, en los que la función incógnita (la solución del problema) es la suma de funciones de interpolación simples, con coeficientes desconocidos. Así, el problema original con infinitos grados de libertad se transforma en un problema con un número

finito de éstos, es decir, la solución del problema sobre todo el dominio se aproxima a partir de un número finito de coeficientes desconocidos. Con esta filosofía, y aplicando el método variacional de Ritz o el método de Galerkin, se obtiene un sistema de ecuaciones algebraicas, que, una vez resuelto, proporciona la solución aproximada del problema [9].

2.3.4. Malla de diferencias finitas

Las mallas de diferencias finitas son mallas estructuradas, y son utilizadas en el método de diferencias finitas, el cual consiste en una aproximación de derivadas parciales por expresiones algebraicas envolviendo los valores de la variable dependiente en un limitado número de puntos seleccionados.

- Como resultado de la aproximación, la ecuación diferencial parcial que describe el problema es reemplazada por un número finito de ecuaciones algebraicas, escritas en términos de los valores de la variable dependiente en puntos seleccionados. Las ecuaciones son lineales si las ecuaciones diferenciales parciales son también lineales.

- El valor de los puntos seleccionados se convierte en las incógnitas, en vez de la distribución espacial continua de la variable dependiente. El sistema de ecuaciones algebraicas debe ser resuelto y puede envolver un número largo de operaciones aritméticas.

- Antiguamente todos estos cálculos eran realizados manualmente, pero actualmente estas operaciones son ejecutadas por medio de un programa computacional [6].

2.4. Archivos en formato m2d

Se utilizó archivos en formato m2d para representar triangulaciones en dos dimensiones, estos archivos guardan la información necesaria para poder modelar una triangulación en dos dimensiones.

Nomenclatura del archivo en formato m2d:

- Un # representa una línea explicativa no necesaria para la representación de un triángulo.
- Una v o V representa un vértice.
- Una t o T representa un triángulo.
- Una n o N representa un triángulo vecino.

2.4.1. Estructura del archivo en formato m2d

Un archivo md2 está compuesto por una configuración que contiene el tipo de triangulación utilizada, la cantidad de vértices de la triangulación, además del alto y ancho de ésta, y la cantidad de triángulos generados.

Para representar una triangulación es necesario tener información de vértices, triángulos, y triángulos vecinos, estos datos los contiene el archivo en formato m2d con la siguiente estructura:

- VÉRTICES: En la primera columna de una línea el programa reconoce una V o v como un vértice, en la segunda columna de la misma línea extrae un identificador de tipo entero para el vértice, y en las últimas dos columnas de esta misma línea extrae las coordenadas (x, y) del vértice en el plano euclidiano.

```
# Vertices
v 1 35.6824 80.7952
v 2 28.9463 25.0104
v 3 11.0213 13.1281
v 4 88.3018 18.1144
v 5 89.8467 10.8085
v 6 9.87667 24.9382
```

Vértices

- TRIÁNGULOS: En la primera columna de una línea el programa reconoce una T o t como un triángulo, en la segunda columna de la misma línea extrae un identificador de tipo entero para el triángulo, y en las últimas tres columnas de esta misma línea extrae tres números, los cuales representan los identificadores de los vértices antes guardados para generar un triángulo con tres vértices.

```
# Triangles
t 1 157 406 30
t 2 162 13  77
t 3 322 143 239
t 4 409 289 251
t 5 368 2   47
t 6 476 4   5
```

Triángulos

- TRIÁNGULOS VECINOS: En la primera columna de una línea el programa reconoce una N o n como un triángulo vecino, en la segunda columna de la misma línea extrae un identificador de tipo entero para el triángulo, y en las últimas tres columnas

de esta misma línea extrae tres números, los cuales representan los identificadores de los triángulos vecinos antes guardados, los cuales representan qué triángulos están al rededor de este triángulo.

```
1 # Neighbors
2 n 1 801 19  802
3 n 2 365 473 23
4 n 3 675 283 639
5 n 4 655 626 808
6 n 5 90  125 298
7 n 6 7   936 255
```

Triángulos vecinos

En este capítulo se abordó la forma de organizar la información mediante estructuras de datos, además de explicar las clases contenedoras estáticas que proporcionan almacenamiento para ítems de datos, los cuales también son estáticas. Se abordó también la forma de representar una triangulación en la malla, su estructura y sus componentes, incluyendo los tipos de mallas y sus funcionalidades dependiendo de su estructura. Además se explicó los métodos de diferencias finitas y los métodos de elementos finitos utilizados en la solución de mallas estructuradas y no estructuradas, según corresponda.

Al final de este capítulo se explicó cómo se componen los archivos en formato m2d, su nomenclatura y además cómo se estructuran.

Ya conociendo la forma de representar la triangulación, los archivos necesarios para cargar la información en las estructuras de datos, podemos avanzar al siguiente capítulo donde se aborda el diseño del software, como se estructura el diseño de éste, sus funcionalidades, algoritmos utilizados, además del flujo de los datos.

Capítulo 3

Diseño del software

3.1. Patrón de diseño del software

Se utilizó el patrón de diseño Modelo-Vista-Controlador (MVC), el cual consta de un modelo de datos, de información de presentación y de información de control, además se agregó un autómata para actualización del dibujo de la vista. Para la utilización de este patrón de diseño se requiere que cada uno de los elementos esté separado en distintos objetos.

Para generar este software se utilizó un Applet el cual por definición significa:

"*Los Applets son pequeñas aplicaciones a las que se accede desde Servidor de Internet, transportado por Internet, instalado automáticamente y ejecutado como parte de un Documento web. Después de que llega un applet al cliente, tiene acceso limitado a los recursos, por lo que puede producir una interfaz de usuario multimedia arbitraria y ejecutar cálculos complejos sin introducir el riesgo de virus o vulnerar la integridad de los datos*" [18].

3.1.1. Flujo de la información

Para entender el flujo de la información de manera global es necesario analizar el diagrama del patrón de diseño (Ver Figura: 3.1). Primero, el usuario utiliza la applet que en este caso corresponde al controlador, después el usuario interactúa mediante los eventos Listener[1] que tiene la clase controlador, después el controlador actualiza los datos a la clase modelo y los emplea para gestionar los dibujos, al tener actualizados los datos del modelo, el controlador envía los datos al gestor de dibujo el cual se encarga de mantener a la vista informada de los cambios generados en el dibujo. Después de enviar los datos al gestor de dibujo, el controlador genera la vista actualizada con la información que el usuario envió al applet (Ver Figura 3.1).

El diagrama de clases del patrón de diseño es necesario para entender la estructura del Modelo-Vista-Controlador, puesto que es la Clase Controlador la que se encarga de gestionar la Vista y los datos del Modelo, un usuario al acceder a una vista del software interactúa con ella, por medio de componentes visuales de javax.swing y estos son ejecutados mediante los Listener que se encuentran en la Clase Controlador, por ultimo la Clase Modelo se encarga de guardar y gestionar los datos necesarios para la representación gráfica y el manejo de los datos para los algoritmos de refinamiento de mallas.

[1]se encargan de controlar los eventos, esperan a que el evento se produzca y realiza una serie de acciones. Según el evento, necesitaremos un Listener que lo controle

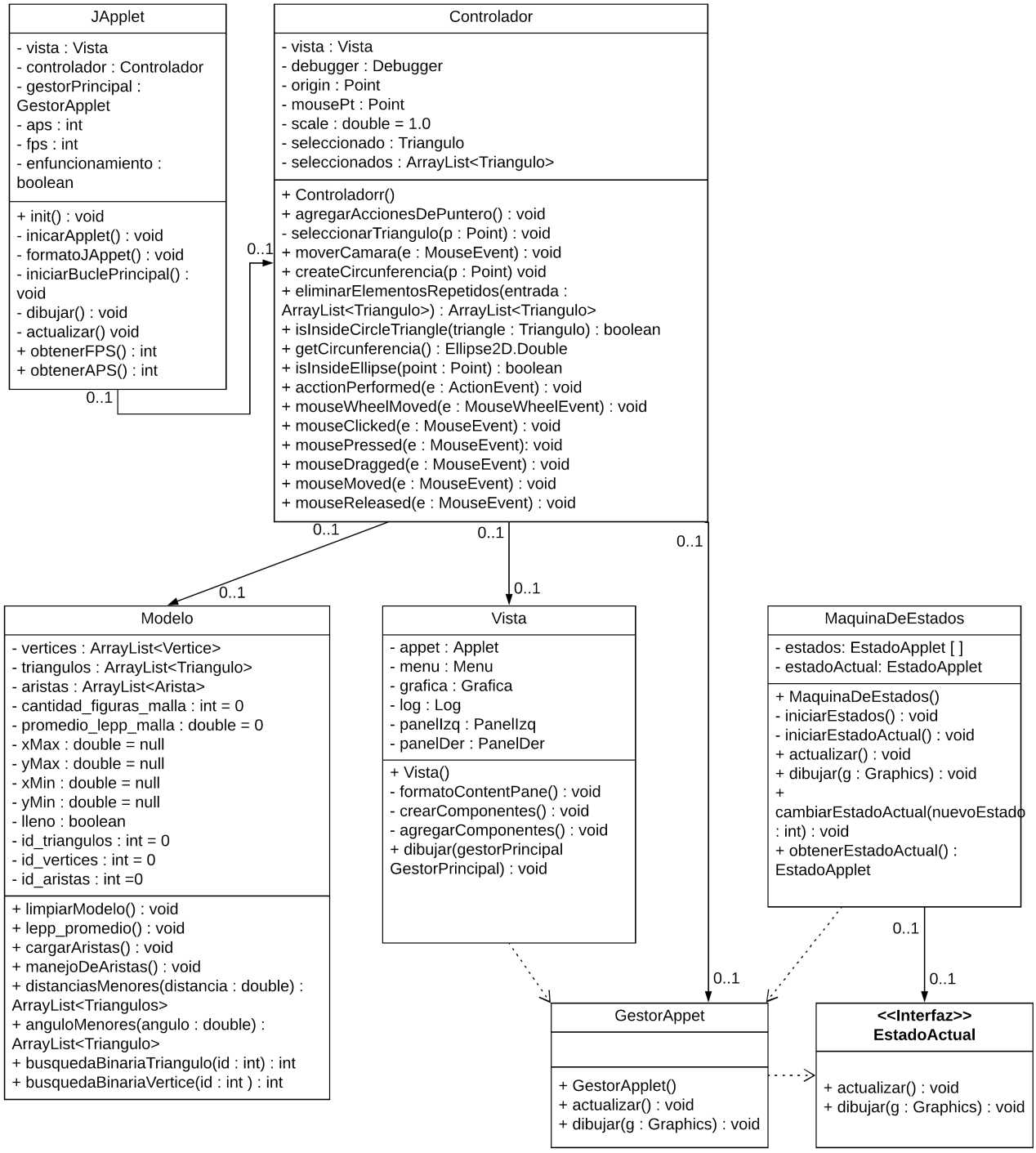


Figura 3.1: Diagrama de clases del patrón de diseño

3.1.2. Diagrama de secuencia

El diagrama de secuencia es utilizado para modelar la interacción entre objetos del sistema. Los diagramas de interacción se encargan de describir el comportamiento dinámico del sistema de información mediante el paso de mensajes entre los objetos del mismo [2].

Componentes :

- Un objeto se representa con un rectángulo de encabezado, con el nombre del objeto en su interior y una línea vertical discontinua, llamada línea de vida (Ver Figura 3.2).

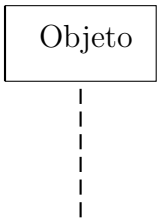


Figura 3.2: Objeto diagrama secuencia

- Un mensaje se representa como una flecha horizontal entre las líneas de vida de los objetos que intercambian el mensaje(Ver Figura 3.3).

Figura 3.3: Mensaje diagrama secuencia

- Una partición representa una serie de pasos o ciclos donde se estructuran mensajes entre objetos (Ver Figura 3.4).

[2] https://es.scribd.com/doc/52627593/Diagramas-para-Diseno-de-Software

Figura 3.4: Partición diagrama secuencia

En nuestro diagrama (Ver Figura: 3.5) el usuario ingresa a la pagina web donde se inicia el applet que en este caso corresponde a la clase controlador, este instancia los objetos modelo, vista, gestorApplet y utiliza el método IniciarApplet(), al iniciar este método se detonan los siguientes métodos:

iniciarControlador Este método es el encargado de configurar el applet, que sea visible que ignore el repintado y que se pueda hacer click en él.

iniciarVista Este método se encarga de configurar el objeto vista, que sea visible, configurar sus dimensiones (alto-ancho), además de dar instancia a los elementos contenidos en la vista, con sus respectivos eventos Listener.

iniciarBuclePrincipal Este método se encarga de refrescar y dibujar el canvas, a través de un ciclo que se refresca cada un segundo, además, al tener el método dibujar de la vista en este bucle, se puede tener una vista informada de los eventos de interacción y de escucha por medio de un autómata, el cual interacciona con la vista y con el controlador. La vista por otro lado está siempre a la espera de interacciones, las cuales puede ser; abrir un modelo, al abrirlo se cargan los datos en el modelo y se lo envía al controlador el cual tiene los objetos principales.

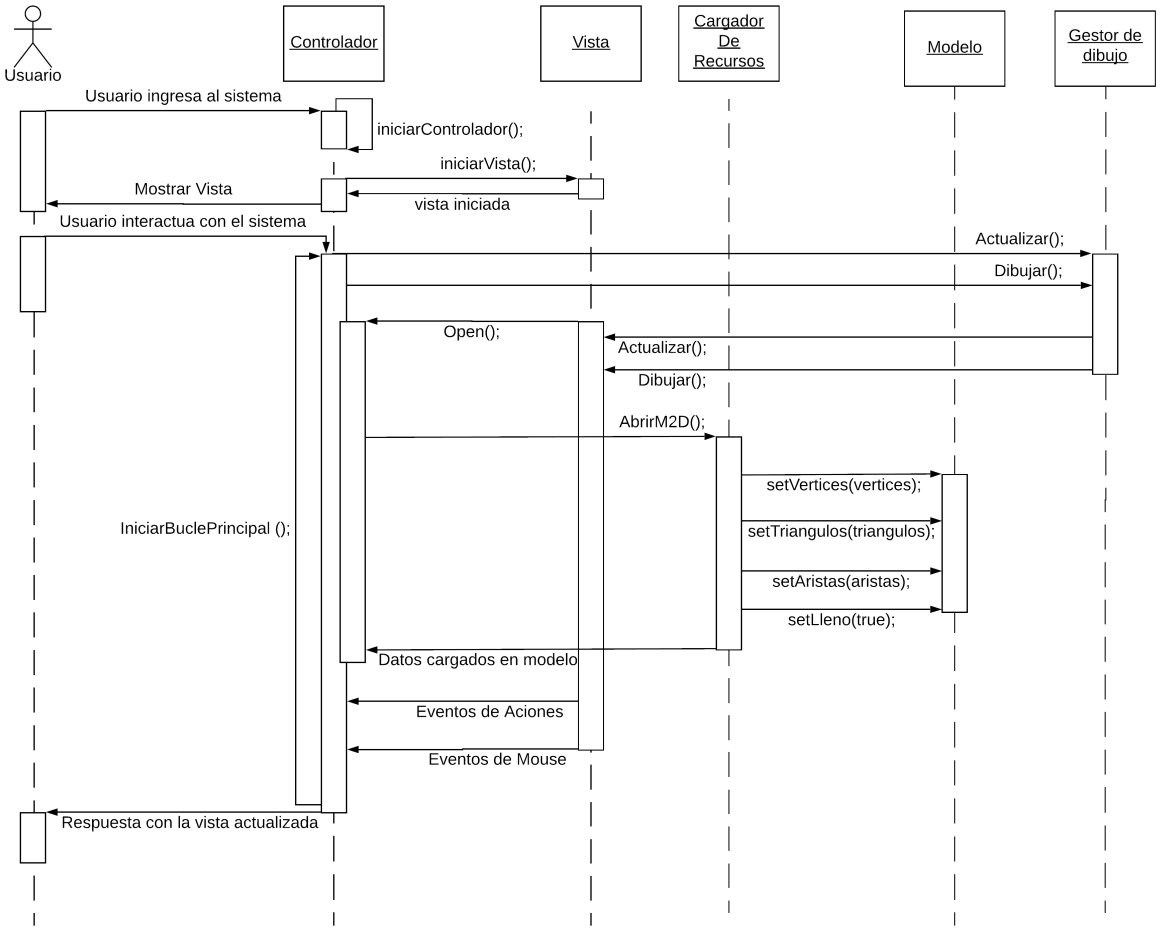


Figura 3.5: Diagrama de secuencia

3.2. Metodología del patrón de diseño

La metodología a utilizar en el patrón de diseño corresponde a dividir para conquistar. A continuación se explica cómo se compone cada elemento del diagrama del patrón de diseño, para así poder entender la estructura detallada del sistema desarrollado.

3.2.1. Clase vista

Nuestra clase vista se compone de cinco elementos o clases (menuDeBarras, Grafica, PanelIzq, PanelDer, Log) las cuales estructuran la vista del applet, además esta clase recibe un Japplet el cual es instanciado en la clase controlador, éste corresponde al Japplet original. El Japplet está compuesto por la clase menuDeBarras y un panel llamado content, el cual contiene las clases Grafica, PanelIzq, PanelDer y Log, como se muestra en la figura 3.6.

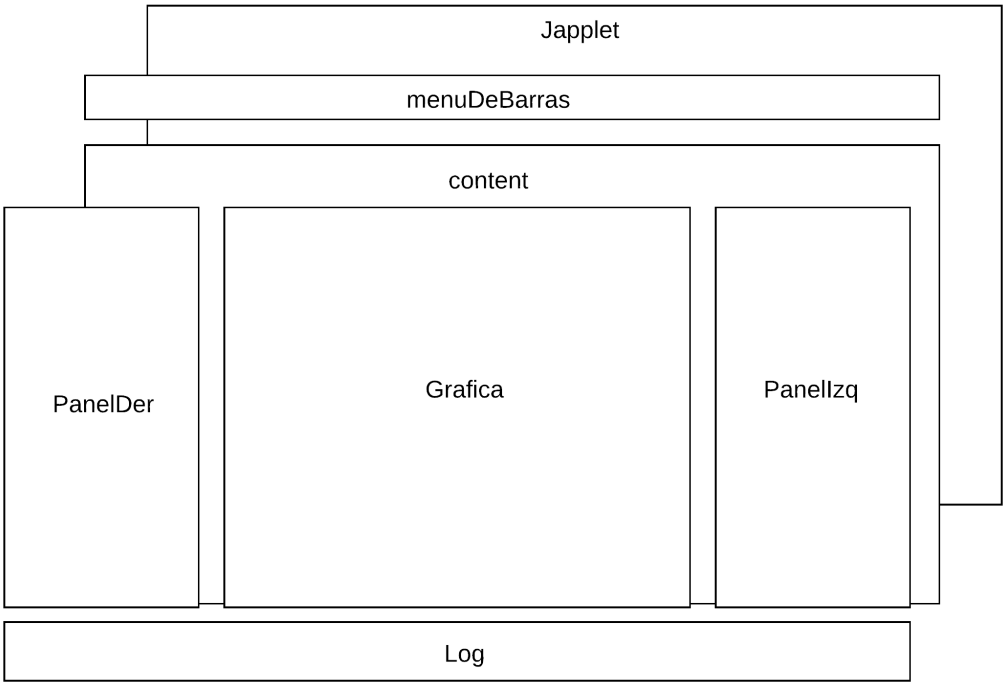


Figura 3.6: Mockup de la estructura de la clase vista

Cada una de los elementos de la vista interactúan entre sí desde la clase controlador, ésta es la encargada de gestionar cada una de las interacciones que el usuario haga en cualquiera de los elementos.

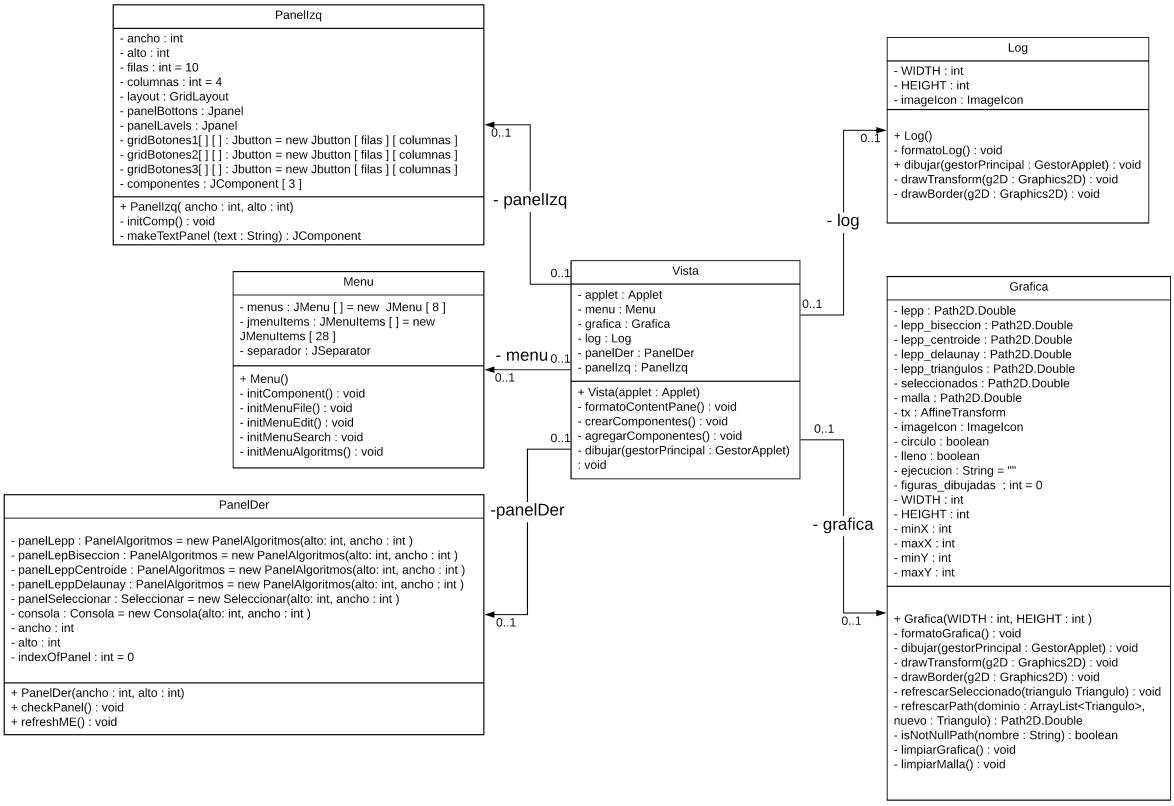


Figura 3.7: Diagrama de clases de la clase vista

La clase vista es una clase contenedora y mediadora entre el controlador y los cinco paneles a los cuales el usuario tiene acceso. Utilizando la metodología divide y vencerás se pudo atomizar la clase vista para así tener una visión detallada y modularizada de cada uno de los elementos que la componen. A continuación se explica cada uno de los elementos que componen la clase vista (Ver Figura 3.7).

3.2.1.1. MenuDeBarras

El menuDeBarras es una clase de selección de menús, primero hay que conocer los JComponent's que esta compuesto el elemento menuDeBarras (Ver Figura 3.8), a continuación se explica cada uno de los componentes utilizados en el diseño del elemento menuDeBarras.

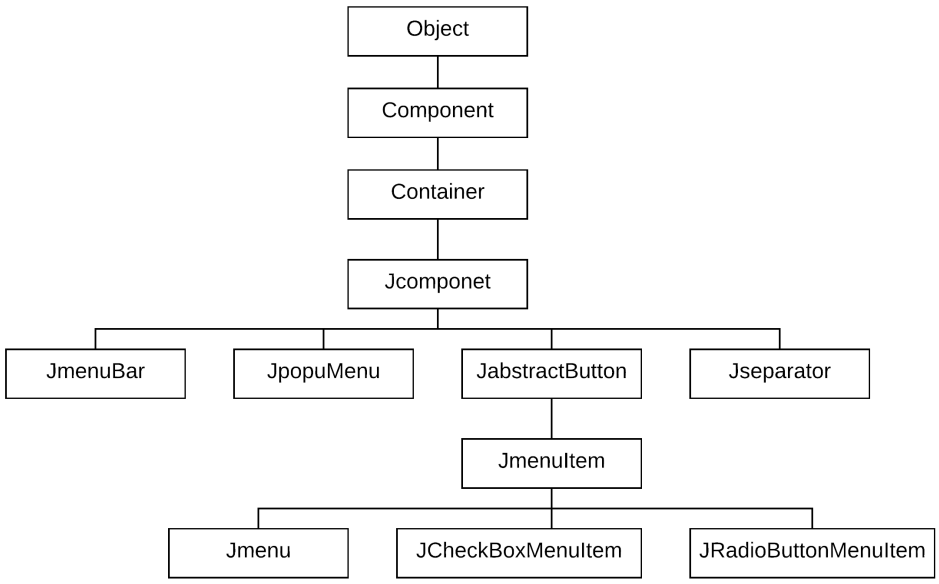


Figura 3.8: Diagrama de JComponent's de la clase menuDeBarras [2]

JmenuBar En nuestro menuDeBarras la misma clase extiende JmenuBar, lo que genera que empaquete todo los posibles elementos que se adicionen, los JmenuBar corresponden a el espacio donde se puede agregar diversos elementos que compondrán el menú, un JmenuBar contiene elementos llamados JMenu (Ver Figura 3.9).

JMenu Un JMenu es un widget que contiene un panel desplegable con JMenuItem (Ver Figura 3.9).

JMenuItem Un JMenuItem corresponde al widget final donde se aplicará la funcionalidad correspondiente (Ver Figura 3.9).

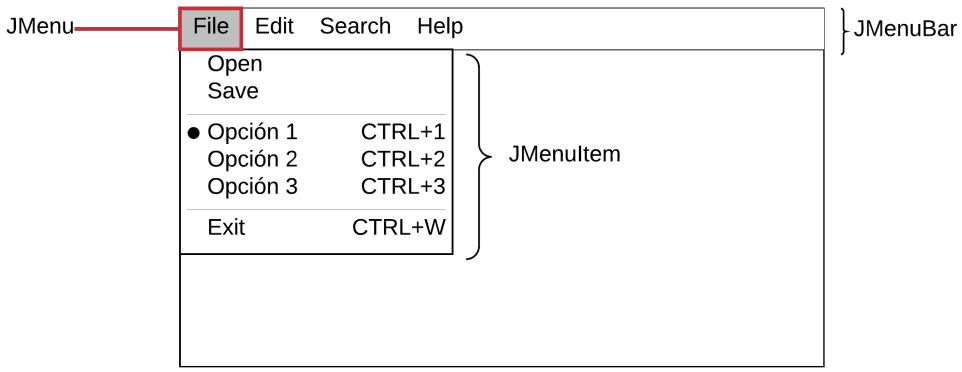


Figura 3.9: Mockup de la clase menuDeBarras

En nuestro Sistema la clase MenuDeBarras se encarga de tener el acceso al cargado y el guardado de datos, además de cerrar el Japplet, aplicar algoritmo de refinamiento de mallas, aplicar algoritmos de mejoramiento de mayas y de la configuración del software, también contiene un menú de ayuda.

3.2.1.2. Gráfica

La clase Gráfica es una clase de pintado y actualización, primero hay que conocer los JComponent's que está compuesto el elemento Gráfica (Ver Figura 3.10), a continuación se explicará cada uno de los componentes utilizados en el diseño del elemento Gráfica.

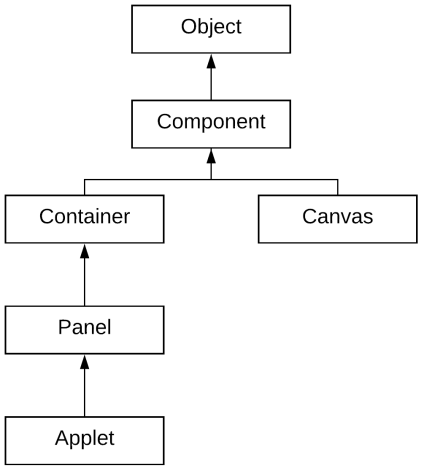


Figura 3.10: Diagrama de JComponent's de la clase gráfica [2]

Canvas En nuestra Gráfica, la misma clase extiende de Canvas, lo que genera que por herencia tenga acceso a los componentes de canvas, esta clase es utilizada para mostrar las mallas no estructuradas y poder interactuar con éstas (Ver Figura 3.10).

Es en esta clase, donde se hace el dibujado, mediante el objeto GestorApplet, el cual ocupa una estrategia de buffer de cuatro espacios de memoria, después de esto el buffer es enviado al objeto de gráficas en dos dimensiones, con el objeto de gráficas en dos dimensiones instanciado se le ingresa un fondo estático, antes de enviarlo al GestorApplet para su actualización global, luego de esto se dibuja en el objeto de gráficas en dos

dimensiones, todo lo dibujado en esta clase se sincroniza con las otras clases que ocupan este objeto instanciado, luego de esto se borra el objeto de gráficas en dos dimensiones, y la estrategia de buffer libera su espacio de memoria (Ver Figura 3.11).

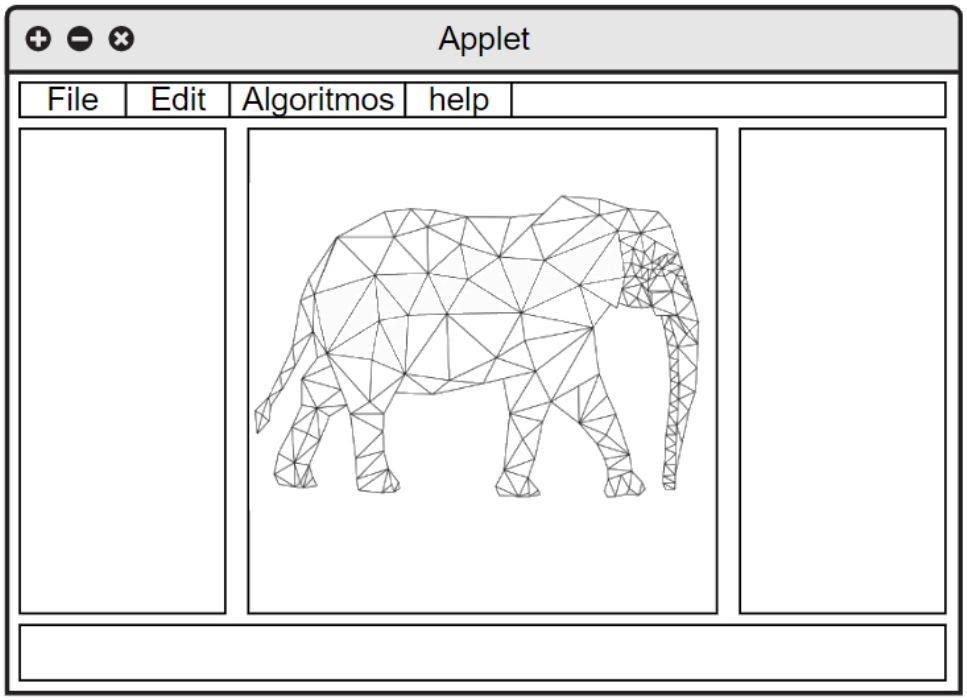


Figura 3.11: Mockup de la clase gráfica

3.2.1.3. PanelIzq

Esta clase es la encargada de tener botones de acceso rápido a las funcionalidades de la Japplet, a continuación de explicara los JComponent's que esta compuesta la clase PanelIzq (Ver Figura 3.12).

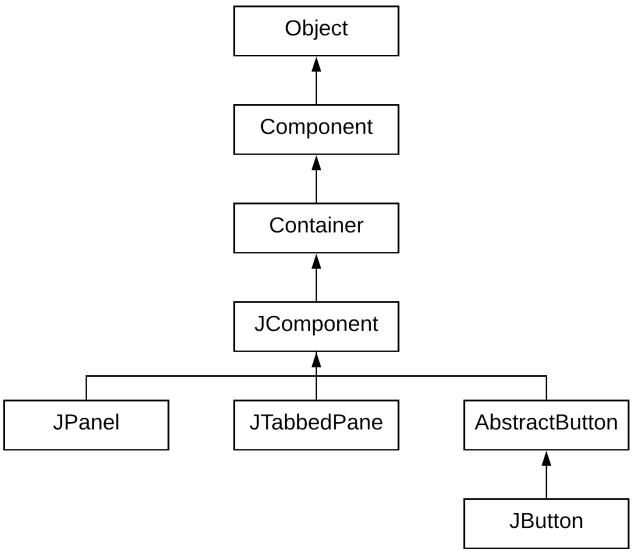


Figura 3.12: Diagrama de JComponent's de la clase panelIzq [2]

JPanel Nuestra misma clase PanelIzq extiende de Jpanel y hereda todos sus componentes, esta clase se utiliza para mostrar de forma gráfica el panel, donde el usuario puede interactuar entre distintas funcionalidades (Ver Figura 3.13).

JTabbedPane El JTabbedPane corresponde a un conjunto de paneles en una misma dimension, en nuestro caso su utilidad es la de separar distintos grupos de botones, dependiendo de la funcionalidad de canda conjunto de botones (Ver Figura 3.13).

JButton Los JButton corresponden a las acciones predeterminadas guardadas en memoria del software, las cuales el usuario que accede a la vista tiene la capacidad de accionarlos (Ver Figura 3.13).

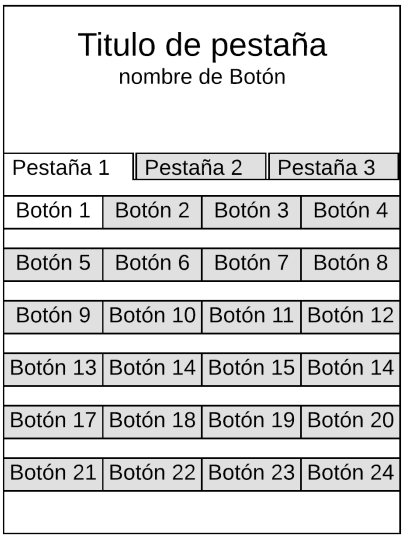


Figura 3.13: Mockup de la clase panelIzq

3.2.1.4. PanelDer

Esta clase es la encargada de mostrar en la información correspondiente a la funcionalidad que se quiera ejecutar, por ejemplo si se desea seleccionar un segmento de la triangulación en este panel se mostrará cada uno de los elementos necesarios para esta selección, como su radio (si es una circunsferencia), el punto céntrico de la circunsferencia en el plano. Dependiendo de qué funcionalidad se quiera acceder, este panel cambiará los campos que contenga el botón seleccionado. A continuación se explica los JComponent's que está compuesta la clase PanelDer (Ver Figura 3.14).

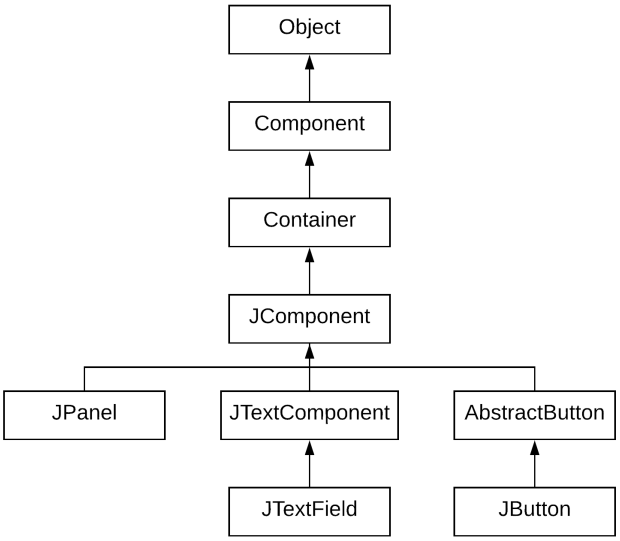


Figura 3.14: Diagrama de JComponent's de la clase panelDer [2]

JPanel Nuestra misma clase PanelDer extiende de Jpanel y hereda todos sus componentes, esta clase se utiliza para mostrar de forma gráfica el panel, donde el usuario puede interactuar entre distintas funcionalidades (Ver Figura 3.15).

JLabel El JLabel corresponde a una etiqueta en la cual se muestra el nombre de los valores correspondientes a cada funcionalidad, un ejemplo de esto es, si se selecciona la funcionalidad circunferencia en el PanelIzq, el PanelDer muestra cada uno de los nombres de los valores necesarios para poder modificar esa circunferencia, ya sea su radio, su punto de origen, etc (Ver Figura 3.15).

JTextField El JTextField corresponde a los campos de entrada de texto, y también muestra el valor que corresponda dependiendo de la etiqueta, además los valores contenidos pueden ser modificados y esto modifica el dibujo en el canvas (Ver Figura 3.15).

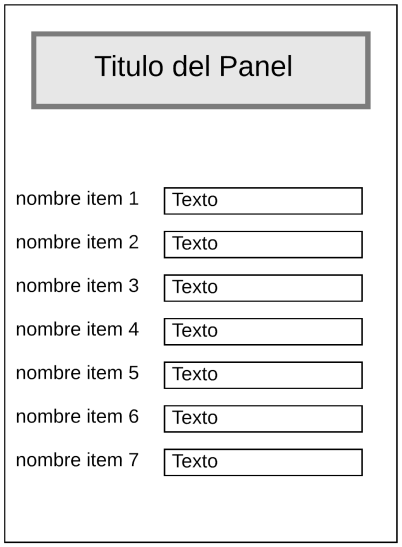


Figura 3.15: Mockup de la clase panelDer

3.2.1.5. Log

Esta clase es la encargada de mostrar en la información correspondiente al registro de información relevante dependiendo del proceso que se ejecuta, por ejemplo si se ejecuta un algoritmo de refinamiento, muestra los datos relevantes a esa funcionalidad, a continuación de explica los JComponent's que está compuesta la clase Log (Ver Figura 3.16).

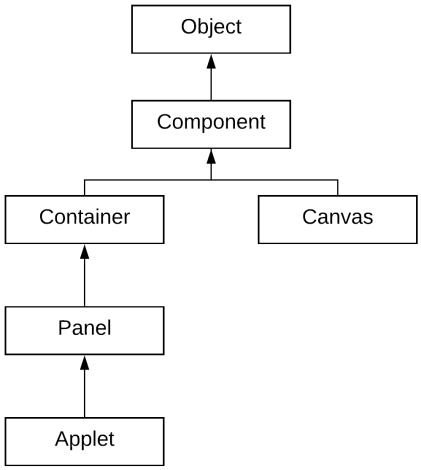


Figura 3.16: Diagrama de JComponent's de la clase log [2]

Canvas En nuestra clase Log, la misma clase extiende de Canvas, lo que genera que por herencia tenga acceso a los componentes de canvas, esta clase es utilizada para mostrar información en tiempo real, puesto que se actualiza en el mismo intervalo que la clase Gráfica, mediante el bucle principal. En esta parte se dibuja, mediante texto, cada uno de los acontecimientos relevantes para el usuario, como por ejemplo cuál es la cantidad de actualizaciones por segundo que tiene la clase Gráfica, la cantidad de frames por segundo que tiene la clase Gráfica, la cantidad de vértices generados por algún refinamiento realizado, el promedio de lepp que la malla contiene, etc (Ver Figura 3.17).

FPS:60 APS:10 Leep:200 triangulos:235 vértices:34565 Texto

Figura 3.17: Mockup de la clase log

3.2.2. Clase controlador

Nuestra clase Controlador se compone de tres elementos o clases (Modelo, Vista, GestorApplet). La clase controlador extiende de JApplet, por lo cual es donde comienza el programa, además es en esta clase donde se manejan todos los eventos de teclado y de mouse.

Esta clase es la encargada de dirigir cada una de las interacciones y funcionalidades del software, cada una de las clases tienen funcionalidades específicas las cuales son ejecutadas en la clase controlador.

Por extender de JApplet y para poder ejecutar nuestro software en un navegador web es necesario implementar los siguientes métodos:

3.2.2.1. init()

Éste método es el que ejecuta navegador web para que inicie el programa, al iniciar este método se instancia un objeto modelo, un objeto vista y un objeto GestorApplet, además se inicia el método iniciarApplet.

iniciarApplet() El método iniciarApplet es el encargado de iniciar los métodos necesarios para que el JApplet defina las características del JApplet y sus funcionalidades, además hace que el atributo en funcionamiento el cual es de tipo boolean pase a estar en true, este atributo es una bandera el cual hace que el método iniciarBuclePrincipal() pueda funcionar, los métodos que contiene la clase iniciarApplet son los siguientes:

- iniciarControlador(): Éste se encarga de hacer que el JApplet no tenga repintado (setIgnoreRepaint(true)), que los componentes integrados en el JApplet puedan tener el foco si es requerido para ello (setFocusable(true)), que el JApplet pueda ser visible (setVisible(true)), y que se pueda hacer llegar el componente foco de entrada (requestFocus()).

- iniciarVista(): Éste se encarga de hacer que la clase Vista sea visible (setVisible(true)), también se encarga de definir las dimensiones de la clase vista (setSize(ancho,alto)), además se encarga de que la clase controlador pueda tener acceso a todos los eventos de la clase Vista, ya sea por teclado o por mouse.

- iniciarBuclePrincipal(): Éste se encarga de hacer que la clase Controlador dirija los métodos de pintado (dibujar()), y de actualización (actualizar()), a través de una iteración la cual termina cuando la bandera en funcionamiento pasa a estar en estado false.

En este ciclo se toma el tiempo cuando entra en el ciclo, después se ve la diferencia entre el tiempo de inicio del ciclo y el tiempo en que el ciclo llega al final, para tener una referencia en el tiempo en que se demora el ciclo en llegar de su inicio a su término. Después de esto, se guarda en otra variable el tiempo de inicio del bucle, el cual es ocupado para poder calcular la diferencia entre un ciclo y otro, más adelante se hace una sumatoria del tiempo transcurrido por cada ciclo del bucle dividido por uno, el cual es utilizado en un blucle while.

Éste bucle solo es ingresado si la sumatoria del tiempo transcurrido es mayor o igual a uno, además se encarga de ejecutar el método actualizar(), también de sumar uno a el atributo **actualizacionesAcumuladas**, para saber cuántas veces se actualiza el programa, y cada vez que entra a este ciclo se le resta uno para que solamente se ingrese una vez. Después de terminar de actualizar el programa, se procede a dibujar y se contabiliza cuantas veces se dibuja en el programa.

Por último, se toma referencia al tiempo actual y se le resta el tiempo transcurrido al término del ciclo principal, y si este es mayor a un segundo, se restablece los contadores de las actualizaciones acumuladas y el contador de dibujos pintados, y se toma el tiempo en que se termina el bucle principal (Ver Figura 3.5).

3.2.3. Clase modelo

Esta clase es la encargada de guardar los datos provenientes del archivo M2d, estos datos son guardados en sus estructuras de datos correspondientes (Vértice, Arista, Triángulo), la finalidad de esta clase es mantener la información en diferentes estructuras.

Cuando se instancia el objeto Modelo, éste, en su constructor, crea una lista de arreglos de vértices, una lista de arreglos de aristas, una lista de arreglos de triángulos, las cuales están vacías, además se instancia una variable de tipo boolean que es para saber si las listas de arreglos contienen información.

Por otro lado esta clase contiene métodos get() y set(), los cuales se ocupan para extraer información de las estructuras contenidas en la clase, además de los siguientes métodos de clase:

3.2.3.1. LimpiarModelo()

El método LimpiarModelo se encarga de que todos los atributos que tiene la clase modelo, se borren y pasen a estar vacíos.

3.2.3.2. MaxMin()

El método MaxMin se encarga de encontrar las coordenadas x, y menores, y las coordenadas x, y mayores, en el conjunto de coordenadas contenidas en el objeto de la clase vértice.

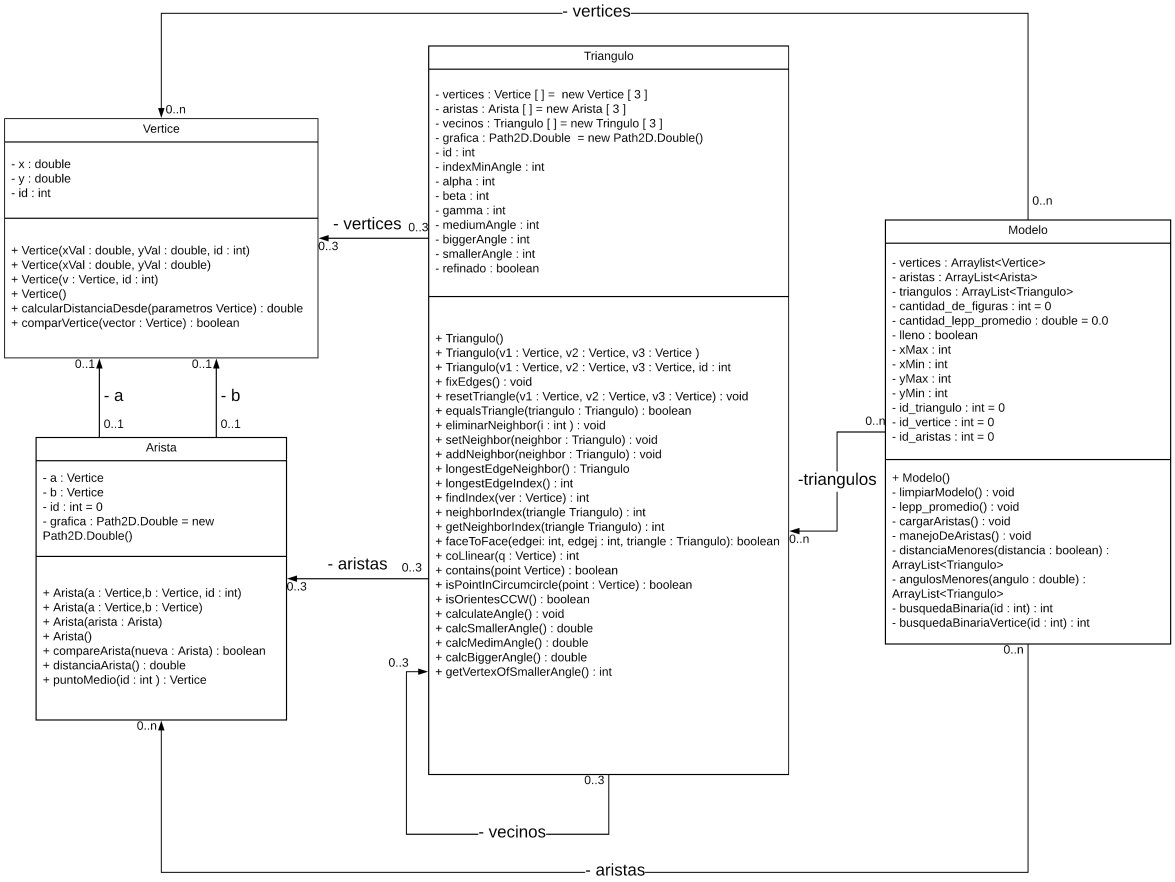


Figura 3.18: Diagrama de clases de la clase modelo

3.3. Casos de uso

3.3.1. Diagrama casos de uso

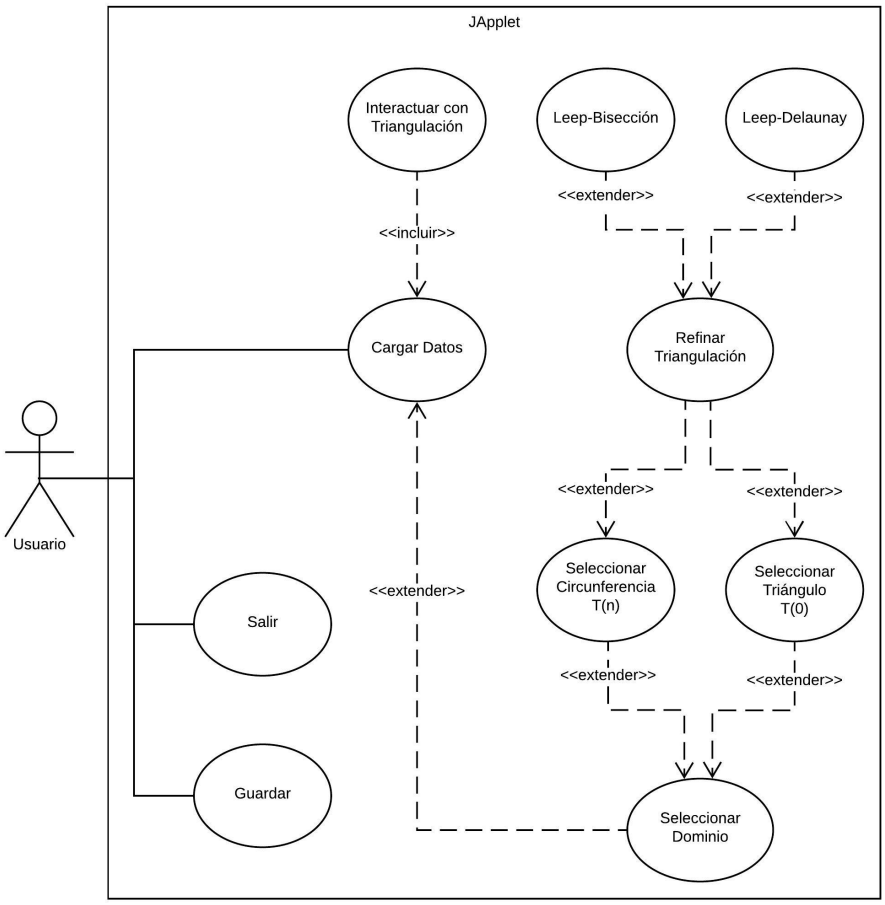


Figura 3.19: Diagrama de casos de uso

3.3.2. Descripción casos de uso

3.3.2.1. caso de uso <guardar>

Nombre	Guardar	**CU1**
Identificador caso de uso	001	
Prioridad	Baja	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del guardado de la triangulación, en un archivo de modelado en dos dimensiones (.md2).	
Pre-Condiciones	El usuario debe cargar al sistema un archivo de modelado en dos dimensiones (.m2d), para que éste cargue los datos en su estructura.	
Flujo de Eventos Básicos		
Descripción	El usuario presiona el botón guardar, el sistema abre una ventana para eligir un archivo, después comprueba si ha presionado el botón aceptar, luego de esto se obtiene la dirección (path) donde se desea guardar el archivo, luego de esto crea el archivo, después se escribe en el archivo los datos relevantes de forma estructurada, se cierra el archivo y se comprueba si al guardar se obtuvo la extension del archivo, si no, se la agrega el sistema, y por último se muestra un mensaje de guardado exitoso.	
Al actor	***El sistema***	
1. Presiona el botón guardar.	**2(a).** Abre una ventana para eligir un archivo.	

3. Selecciona una dirección (path) donde se desea guardar el archivo y presiona el botón aceptar .	**4.** Comprueba si ha presionado el botón aceptar, después se obtiene la dirección (path) donde se desea guardar el archivo. Se escribe en el archivo los datos que se encuentran en el modelo (aristas, vértices , triángulos) de forma estructurada y se cierra el archivo, por último se comprueba si al guardar se obtuvo la extension del archivo, si no, se la agrega el sistema y muestra un mensaje de guardado exitoso.
Flujo de Eventos Alternativo	
Descripción	El usuario presiona el botón guardar y el sistema no tiene una triangulación cargada en memoria temporal, como respuesta el sistema notifica que no se puede guardar, puesto que no se encuentra una triangulación ingresada.
Al actor	***El sistema***
1. Presiona el botón guardar.	**2(b).** notifica que no se puede guardar, puesto que no se encuentra una triangulación ingresada.
Post-Condiciones	No presenta
Nivel	bajo

Tabla 3.1: Descripción caso de uso <guardar>

3.3.2.2. caso de uso <salir>

Nombre	Salir	**CU2**
Identificador caso de uso	002	
Prioridad	Baja	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón salir, el cual se ocupa para cerrar la aplicación.	

Pre-Condiciones	El usuario debe haber ingresado a la aplicación.
Flujo de Eventos Básicos	
Descripción	El usuario presiona el botón salir, y el sistema termina de hacer funcionar la aplicación.
Al actor	***El sistema***
1. Presiona el botón salir.	**2.** Ejecuta una linea de comando para cerrar la aplicación.
Flujo de Eventos Alternativo	
Descripción	no presenta.
Al actor	***El sistema***
No presenta.	No presenta.
Post-Condiciones	No presenta
Nivel	bajo

Tabla 3.2: Descripción caso de uso <salir>

3.3.2.3. caso de uso <cargar datos>

Nombre	Cargar Datos	**CU3**
Identificador caso de uso	003	
Prioridad	alta	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón Abrir, el cual se ocupa para cargar archivos de modelado en dos dimensiones (.m2d), en el modelo de datos y sus estructuras de datos correspondientes.	
Pre-Condiciones	El usuario debe haber ingresado a la aplicación.	
Flujo de Eventos Básicos		

Descripción	El usuario presiona el botón abrir, y el sistema pregunta si esta cargado el modelo de datos, si este esta cargado con información, el sistema se encarga de borrar esta información, luego de esto el sistema instancia las estructuras de datos correspondientes al modelo de datos y procede a cargar los datos en memoria temporal, primero despliega una ventana de dialogo donde el usuario se encarga de encontrar el archivo de modelado en dos dimensiones (.m2d), luego el sistema verifica si el usuario selecciono el botón aceptar, si lo selecciono procede a cargar en memoria temporal el archivo seleccionado, para cargar en memoria temporal el archivo seleccionado primero se lee el archivo, al tener el archivo lo ingresa a la memoria de almacenamiento temporal de información para luego de esto leer cada linea del archivo e ingresarlo a su estructura correspondiente, luego de pasar los datos desde la memoria de almacenamiento temporal hacia la estructura de modelado, se libera la memoria ocupada. Luego de cargar en el modelo los datos, informa a la vista que el modelo esta listo para ser visualizado.
Al actor	***El sistema***
1. Presiona el botón abrir.	**2(a).** Pregunta si esta cargado el modelo de datos, si no esta cargado con información, el sistema instancia las estructuras de datos correspondientes al modelo de datos, después despliega una ventana de dialogo para buscar el archivo de modelado en dos dimensiones (.m2d)

3. Busca el archivo de modelado en dos dimensiones (.m2d) y presiona el botón abrir de la ventana desplegada.	**4.** Verifica si el usuario selecciono el botón aceptar, si lo selecciono crea un archivo para la lectura en el cual carga los datos del archivo seleccionado, y crea un objeto de lectura de archivos, se pasa el objeto de lectura de archivos a la memoria de almacenamiento temporal de información, desde esta se lee cada linea del archivo y lo ingresa a su estructura correspondiente (vértice, arista, triángulo), se libera la memoria ocupada y se informa a la vista que el modelo esta listo para ser visualizado
Flujo de Eventos Alternativo	
Descripción	no presenta.
Al actor	***El sistema***
1. Presiona el botón abrir.	**2(b).** Pregunta si esta cargado el modelo de datos, si este esta cargado el modelo con información, el sistema se encarga de borrar esta información, después se despliega una ventana de dialogo para buscar el archivo de modelado en dos dimensiones (.m2d),
3. Busca el archivo de modelado en dos dimensiones (.m2d) y presiona el botón abrir de la ventana desplegada.	**4(b).** Verifica si el usuario selecciono el botón aceptar, si selecciono cancelar o cerrar la ventana desplegada deja en null el objeto donde se carga el archivo y deja null el objeto de lectura de archivos, termina la ejecución del botón.
Post-Condiciones	No presenta
Nivel	medio

Tabla 3.3: Descripción caso de uso <cargar datos>

3.3.2.4. caso de uso <interactuar con la triangulación>

Nombre	Interactuar con la triangulación	**CU4**
Identificador caso de uso	004	
Prioridad	Baja	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad entre la interación entre el usuario y el lugar donde se visualiza la triangulación (canvas), en este caso, se puede hacer zoom, y mover la cámara hacia la izquierda, derecha, arriba, abajo.	
Pre-Condiciones	El usuario debe haber ingresado una triangulación para poder visualizarla.	
Flujo de Eventos Básicos		
Descripción	El usuario hace click izquierdo en una parte de la visualización de la triangulación y deja el presionado y puede mover la triangulación hacia arriba, abajo, izquierda, derecha.	
Al actor	***El sistema***	
1(a). Deja presionado el el click izquierdo en la visualización triangulación.	**2(a).** Primero se consulta si donde se esta presionando es la parte gráfica del software.	
3. Puede mover el puntero hacia arriba, abajo, izquierda o derecha para desplazar la triangularización.	**4.** Captura las coordenadas x, y del evento, las cuales corresponden a la parte gráfica del software, después actualiza el origen de la gráfica con las nuevas coordenadas capturadas por el evento del puntero, al modificar el origen, se actualiza un atributo de la vista, y al repintar la parte gráfica de la vista, esta actualiza la malla de triángulos desplazada con las coordenadas tomadas por el puntero, al hacer todo esto dentro del bucle principal, se percibe una fluidez en el movimiento de la trangulación, por último se hace un repintado de la gráfica.	

Flujo de Eventos Alternativo	
Descripción	El usuario posiciona el puntero en la arte gráfica y ocupa la rueda de desplazamiento para hacer un acercamiento (zoom in) o un retroceso (zoom out) en la triangulacion.
Al actor	***El sistema***
1(b). Posiciona el puntero al interior de la parte gráfica del software y mueve hacia arriba o abajo la rueda de desplazamiento (scroll wheel).	**2(b).** Primero se verifica si el puntero se encuentra en la parte grafica del software, segundo verifica si el evento del puntero es de tipo "MouseWheelEvent", con esto genera una escala con un valor delta multiplicado por la cantidad de rotaciones de la rueda de desplazamiento, después elije el mayor valor entre la escala y el delta, tras esto captura las coordenadas *x, y* del evento del puntero, después modifica el atributo de la vista de tipo "AffineTransform" para que las nuevas coordenadas ingresadas por el puntero se puedan escalar por su valor absoluto y por último revalida la gráfica y la repinta
Post-Condiciones	No presenta
Nivel	Alto

Tabla 3.4: Descripción caso de uso <interactuar con la triangulación>

3.3.2.5. caso de uso <seleccionar dominio>

Nombre	Seleccionar Dominio	**CU5**
Identificador caso de uso	005	
Prioridad	baja	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón seleccionar Dominio, el cual se ocupa para acceder al menú de selección de dominios.	

Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación.
Flujo de Eventos Básicos	
Descripción	El usuario presiona el botón seleccionar Dominio, el cual le muestra en un panel todos los tipos de selección de dominios que cuenta el software.
Al actor	***El sistema***
1. Presiona el botón seleccionar Dominio.	**2.** Muestra el conjunto de botones que cuenta el panel de selección de dominio.
Flujo de Eventos Alternativo	
Descripción	no presenta.
Al actor	***El sistema***
No presenta.	No presenta.
Post-Condiciones	No presenta
Nivel	bajo

Tabla 3.5: Descripción caso de uso <seleccionar dominio>

3.3.2.6. caso de uso <seleccionar circunferencia>

Nombre	Seleccionar Circunferencia	**CU6**
Identificador caso de uso	006	
Prioridad	alta	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón seleccionar circunferencia, el cual se ocupa para marcar en un circunferencia el conjunto de n triángulos contenidos en una circunferencia.	
Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación y debe haber accedido al menú de selección de dominio.	
Flujo de Eventos Básicos		

Descripción	El usuario presiona el botón "Circunferencia", luego de esto mueve el puntero en la parte de la visualización de la triangulación (grafica) y hace click derecho y deja lo presionado para luego mover el puntero para aumentar o reducir el radio de la circunferencia pintada en la visualización de la triangulación (grafica).
Al actor	***El sistema***
1. Presiona el botón Circunferencia.	**2.** Verifica si el botón seleccionado es el que corresponde a el botón Circunferencia. Borra cualquier pintado existente sobre el pintado de la triangularización y muestra en el panelder, el detalle del radio de la circunferencia generada, además de los puntos *x, y* del centro de la circunferencia. Por último informa a la vista que esta preparada una circunferencia para pintarla sobre la triangulación y modifica los valores necesarios para generar una circunferencia.
3. Selecciona un área.	**4.** Obtiene los valores del área seleccionada y los envía a la vista para su pintado.
5. Mueve el área seleccionada.	**6.** Obtiene los valores del área en desplazamiento y los envía a la vista para su pintado.
Flujo de Eventos Alternativo	
Descripción	no presenta.
Al actor	***El sistema***
No presenta.	No presenta.
Post-Condiciones	Guarda el área seleccionada para su refinamiento
Nivel	alto

Tabla 3.6: Descripción caso de uso <seleccionar circunferencia>

3.3.2.7. caso de uso <seleccionar Triángulo>

Nombre	Seleccionar Triángulo	**CU7**
Identificador caso de uso	007	
Prioridad	alta	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón seleccionar triángulo, el cual se ocupa para marcar un triángulo de la triangularización.	
Pre-Condiciones	El usuario debe haber cargado los datos en la aplicación y debe haber accedido al menú de selección de dominio.	
Flujo de Eventos Básicos		
Descripción	El usuario presiona el botón "Triángulo", luego de esto mueve el puntero en la parte de la visualización de la triangulación (grafica) y selecciona un triángulo.	
Al actor	***El sistema***	
1. Presiona el botón Triángulo.	**2.** Verifica si el botón seleccionado es el que corresponde a el botón Triángulo. Borra cualquier pintado existente sobre el pintado de la triangularización y muestra en el panelder, el detalle del triángulo seleccionado.	
3. Selecciona un triángulo de la triangularización.	**4.** Identifica que triángulo fue seleccionado y obtiene los valores del triángulo seleccionado, para luego pintarlo en la visualización de la triangulación (gráfica).	
Flujo de Eventos Alternativo		
Descripción	no presenta.	
Al actor	***El sistema***	
No presenta.	No presenta.	
Post-Condiciones	Guarda el triángulo seleccionado para su refinación	
Nivel	alto	

Tabla 3.7: Descripción caso de uso <seleccionar triángulo>

3.3.2.8. caso de uso <refinar triangulación>

Nombre	Refinar Triangulación	**CU8**
Identificador caso de uso	008	
Prioridad	baja	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón refinar triangulación, el cual se ocupa para acceder al menú de selección de refinamiento.	
Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación, también haber seleccionado un dominio para su refinamiento.	
Flujo de Eventos Básicos		
Descripción	El usuario presiona el botón refinar triangulación, el cual le muestra en un panel todos los tipos de refinamientos que cuenta el software.	
Al actor	***El sistema***	
1. Presiona el botón refinar triangulación.	**2.** Muestra el conjunto de botones que cuenta el panel de selección de refinamiento.	
Flujo de Eventos Alternativo		
Descripción	no presenta.	
Al actor	***El sistema***	
No presenta.	No presenta.	
Post-Condiciones	No presenta	
Nivel	bajo	

Tabla 3.8: Descripción caso de uso <refinar triangulación>

3.3.2.9. caso de uso <LEEP-Bisección>

Nombre	LEEP-Bisección	**CU9**
Identificador caso de uso	009	

Prioridad	alta
Actor Principal	Usuario
Descripción	Este caso de uso describe la funcionalidad del botón LEEP-Bisección, el cual se ocupa para hacer un refinamiento LEEP-Bisección.
Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación, también haber seleccionado un dominio para su refinamiento y haber seleccionado el panel de refinamiento.
Flujo de Eventos Básicos	
Descripción	El usuario presiona el botón LEEP-Biseccion, el cual refina el dominio seleccionado anteriormente.
Al actor	***El sistema***
1. Presiona el botón LEEP-Bisección.	**2(a).** Refina el dominio seleccionado y muestra en el panelDer, el cual es el panel de detalle del refinamiento la información esencial.
Flujo de Eventos Alternativo	
Descripción	El usuario presiona el botón LEEP-Bisección y el dominio seleccionado no cumple los requisitos para que el algoritmo funcione.
Al actor	***El sistema***
1. Presiona el botón LEEP-Bisección.	**2(b).** Muestra un mensaje en el panel de detalle de LEEP-Bisección, en el cual se le informa al usuario que el dominio seleccionado no se puede refinar con el algoritmo actual.
Post-Condiciones	Devuelve una triangulación refinada, la cual es cargada y repintada al terminar de refinar.
Nivel	alto

Tabla 3.9: Descripción caso de uso <LEEP-Bisección>

3.3.2.10. caso de uso <LEEP-Centroide>

Nombre	LEEP-Centroide	**CU10**
Identificador caso de uso	010	
Prioridad	alta	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón LEEP-Centoide, el cual se ocupa para hacer un refinamiento LEEP-Centroide.	
Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación, también haber seleccionado un dominio para su refinamiento y haber seleccionado el panel de refinamiento.	
Flujo de Eventos Básicos		
Descripción	El usuario presiona el botón LEEP-Centroide, el cual refina el dominio seleccionado anteriormente.	
Al actor	***El sistema***	
1. Presiona el botón LEEP-Centroide.	**2(a).** Refina el dominio seleccionado y muestra en el panelDer, el cual es el panel de detalle del refinamiento la información esencial.	
Flujo de Eventos Alternativo		
Descripción	El usuario presiona el botón LEEP-Centroide y el dominio seleccionado no cumple los requisitos para que el algoritmo funcione.	
Al actor	***El sistema***	
1. Presiona el botón LEEP-Centroide.	**2(b).** Muestra un mensaje en el panel de detalle de LEEP-Centroide, en el cual se le informa al usuario que el dominio seleccionado no se puede refinar con el algoritmo actual.	
Post-Condiciones	Devuelve una triangulación refinada, la cual es cargada y repintada al terminar de refinar.	
Nivel	alto	

Tabla 3.10: Descripción caso de uso <LEEP-Centroide>

3.3.2.11. caso de uso <LEEP-Delaunay>

Nombre	LEEP-Delaunay	**CU11**
Identificador caso de uso	011	
Prioridad	alta	
Actor Principal	Usuario	
Descripción	Este caso de uso describe la funcionalidad del botón LEEP-Delaunay, el cual se ocupa para hacer un refinamiento LEEP-Delaunay.	
Pre-Condiciones	El usuario debe haber cargado los datos de un archivo de moldeamiento en dos dimensiones (.m2d), en la aplicación, también haber seleccionado un dominio para su refinamiento y haber seleccionado el panel de refinamiento.	
Flujo de Eventos Básicos		
Descripción	El usuario presiona el botón LEEP-Delaunay, el cual refina el dominio seleccionado anteriormente.	
Al actor	***El sistema***	
1. Presiona el botón LEEP-Delaunay.	**2(a).** Refina el dominio seleccionado y muestra en el panelDer, el cual es el panel de detalle del refinamiento la información esencial.	
Flujo de Eventos Alternativo		
Descripción	El usuario presiona el botón LEEP-Delaunay y el dominio seleccionado no cumple los requisitos para que el algoritmo funcione.	
Al actor	***El sistema***	
1. Presiona el botón LEEP-Delaunay.	**2(b).** Muestra un mensaje en el panel de detalle de LEEP-Delaunay, en el cual se le informa al usuario que el dominio seleccionado no se puede refinar con el algoritmo actual.	

Post-Condiciones	Devuelve una triangulación refinada, la cual es cargada y repintada al terminar de refinar.
Nivel	alto

Tabla 3.11: Descripción caso de uso <LEEP-Delaunay>

3.3.3. Actores

Usuario : El actor usuario es el único que tiene acceso al sistema y tiene las siguientes funcionalidades:

- Ver triangularización
- Seleccionar Dominio
- Refinar triangularización
- Guardar triangularización
- Abrir triangularización

En este capítulo se abordó el patrón de diseño utilizado y su metodología, en la creación del software, además se explicó el flujo de la información por medio de un diagrama de secuencia, sus componentes y como se estructuran en el software mediante diagramas de clases.

Por último se explicó la funcionalidad del sistema mediante los diagramas de clases de uso y su descripción, conociendo el diseño del software y su funcionalidad podemos adentrarnos en el siguiente capitulo donde se define que es un refinamiento de mallas, en que consiste la bisección de triángulos y los algoritmos implicados en el desarrollo de la aplicación.

Capítulo 4

Refinamiento de triangulaciones

4.1. Refinamiento de triangulaciones mediante la bisección de triángulos

La finalidad es refinar triangulaciones (insertando nuevos puntos en la triangulación) no estructuradas conformes (donde la intersección entre de triángulos adyacentes es entre un vértice común o una arista común), donde estos algoritmos mantienen la calidad de la triangulacion inicial [14].

Se utilizaron algoritmos de refinamiento de triangulaciones basados en la bisección de triángulos, por la arista más larga, la cual conserva la calidad de la triangularización de entrada. Esto debido a la bisección de triángulos por la arista más larga y a la estrategia de propagación por la arista mas larga (LEPP). Es necesario destacar que las triangulaciones producidas por la bisección de triángulos en general no son Delaunay[10, 11].

4.1.1. LEPP (longest edge propagating path)

Dado un triángulo t_0 a refinar, el LEPP de t_0 (LEPP(t_0)) es una secuencia de triángulos $t_0, t_1, ..., t_{n-1}, t_n$, donde el triángulo t_i (i = 1,2,...,n-1,n) es adyacente al triángulo t_{i-1}, por la arista más larga de t_{i-1}. El LEPP(t_0) es una secuencia finita, y termina con uno o dos triángulos terminales.(Ver Figura 4.1) [11, 12, 14].

Se cumplen las siguientes propiedades de *Longest edge propagation path* en una triangulación [13]

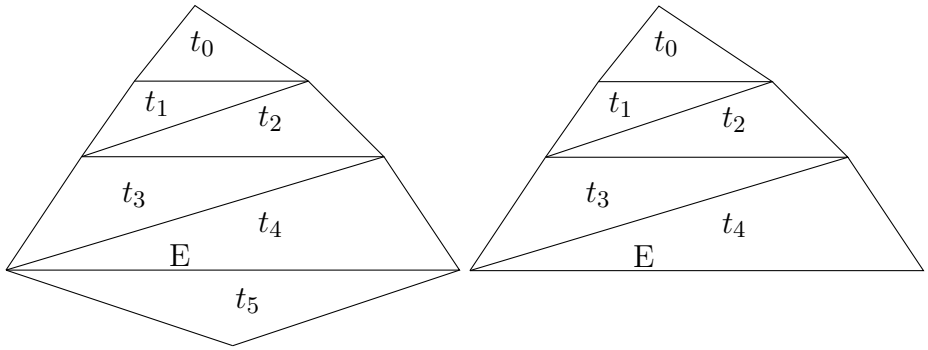
1. Para cualquier triángulo (t), LEPP(t) es finito (puede ocurrir que en el peor de los casos LEPP(t) sea toda la triangulación).

2. Los triángulos $t_0, t_1, ..., t_{n-1}, t_n$, de LEPP($t_0$), tienen su arista mayor en orden estrictamente creciente.

3. Para el triángulo t_n, perteneciente a LEPP(t_0), se cumple una de las siguientes afirmaciones.

(a) La arista más larga de t_n es la misma que la arista más larga de t_{n-1} (arista terminal). (Ver Figura 4.1(a))

(b) Triángulo t_n posee su arista mayor en el borde y dicha arista es mayor que la arista mas larga de t_{n-1} (arista terminal de borde). (Ver Figura 4.1(b))



(a) (b)

Figura 4.1: **(a)** LEPP(t_0) con arista terminal E compartida por los triángulos terminales t_4 y t_5.
(b) LEPP(t_0) con arista terminal de borde E, con un solo triángulo terminal[1].

4.1.1.1. Arista terminal

En una triangulación, una arista terminal, E es compartida por dos triángulos, los cuales son vecinos por esa arista mas larga. Además E, es una arista interior de la triangulación. Los triángulos que comparten la arista terminal son llamados triángulos terminales.

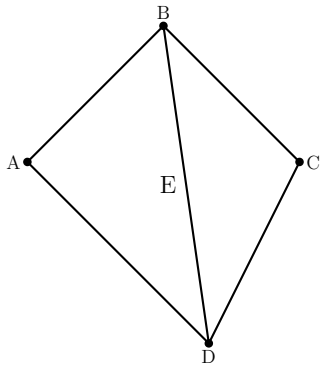


Figura 4.2: Arista terminal (E) de los triángulos terminales ADB y BDC

[1] http://www.repositorio.uchile.cl/handle/2250/102205

4.1.2. Bisección de un triángulo por la arista más larga

Por definición, sea △ABC un triángulo de vértices A, B y C. El proceso de bisección de un △A,B,C por la arista mas larga se define de la siguiente forma [15] :

1. Primero localizar la arista más larga en △ ABC, donde A^i se define como la arista más larga.

2. Luego $D = \frac{V_0(A^i)+V_1(A^i)}{2}$ es el punto medio de la arista más larga, con V_0 y V_1 sus vértices.

3. Construir dos nuevos triángulos a partir del punto medio D. El primer triángulo $\triangle V_0(A^i)DV_2$ y el segundo $\triangle DV_1(A^i)V_2$, donde V_2 es el vértice opuesto a la arista más larga V_0V_1

4. Dado un triángulo $\triangle_X$, con el ángulo interior más pequeño $\alpha > 0$. Bisectar $\triangle_X$ en dos nuevos triángulos, $\triangle_{1i}, i = 1, 2$.

5. Por último Bisectar cada triángulo $\triangle_{1i}$, para formar cuatro nuevos triángulos $\triangle_{2i}, i = 1, 2, 3, 4$, para formar una secuencia "T", de triángulos. Si un $\triangle \in T$, donde "T" es una triangulación conforme, y α es cualquier ángulo interior de $\triangle$, entonces $\alpha \geqslant \frac{\alpha}{2}$ o $\frac{\alpha_0}{2}$ donde α_0 es el ángulo más pequeño de la triangulación inicial T_0

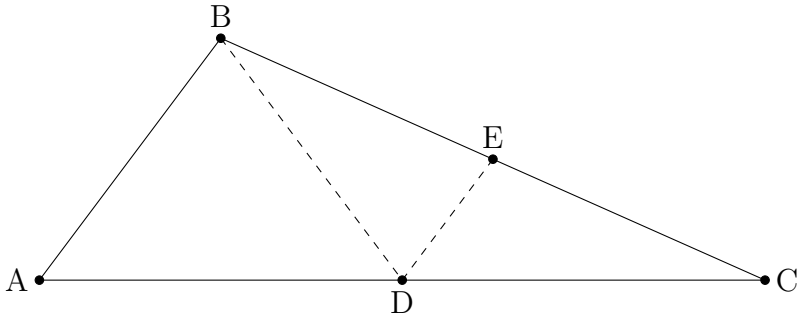


Figura 4.3: Bisección de un triangulo ABC [15]

En la figura 4.3 se muestra la bisección del triángulo ABC por su arista mas larga. Se inserta un punto D en el punto medio de la arista mas larga AC y se traza una nueva arista BD, la cual se extiende desde el punto D, hasta el vértice opuesto B, y esto genera que se formen dos nuevos triángulos, primero $\triangle ADB$ y el segundo $\triangle BDC$.
Luego el triángulo $\triangle BDC$, es bisectado en su arista mas larga BC, se inserta un punto E, en el punto medio de BC, para trazar una arista desde el punto E. hasta el vértice opuesto D, y esto genera dos nuevos triángulos, primero $\triangle BDE$ y el segundo $\triangle DCE$.

4.1.2.1. Conformidad de una malla por la bisección de un triángulo

Sea t_0 un triángulo de una malla conforme "T", en la bisección de t_0 por la arista más larga, hace que la malla "T" quede no conforme en el caso de existir un triángulo vecino t_1 que comparta con t_0 la arista recientemente dividida, debido al punto insertado en la mitad de la arista compartida. De otro modo si la arista más larga del triangulo t_0 pertenece al contorno de la malla, entonces la malla "T" permanecerá conforme [**?**].

4.1.3. Algoritmos de refinamiento de triangulaciones

4.1.3.1. LEPP-bisección

LEPP-Bisección es un algoritmo de refinamiento de triangulaciones por la arista más larga, se ilustra en la figura 4.4, en la cual se muestra el refinamiento del triángulo t_0 sobre la triangulación inicial 4.4(a).

Las figuras 4.4(b),4.4(c),4.4(d),4.4(e), representan los pasos por cada iteración del algoritmo, con la inserción de puntos medios en la arista más larga de cada triángulo (1,2,3,4), hasta llegar a t_0 y bisectarlo, dando como resultado la triangularización refinada 4.4(f).

Algoritmo 1: LEPP-Bisección(t_0)

Entrada: t_0: Triángulo inicial para el refinamiento.

```
1 mientras t_0 no esté bisectado hacer
2 |   t_n ← El último triángulo de LEPP(t_0);
3 |   si t_n está en el borde entonces
4 |   |   Bisectar t_0 por el lado mas largo
5 |   en otro caso
6 |   |   Bisectar por el lado mas largo el último par de triángulos t_n y t_{n-1}
7 |   fin
8 fin
```

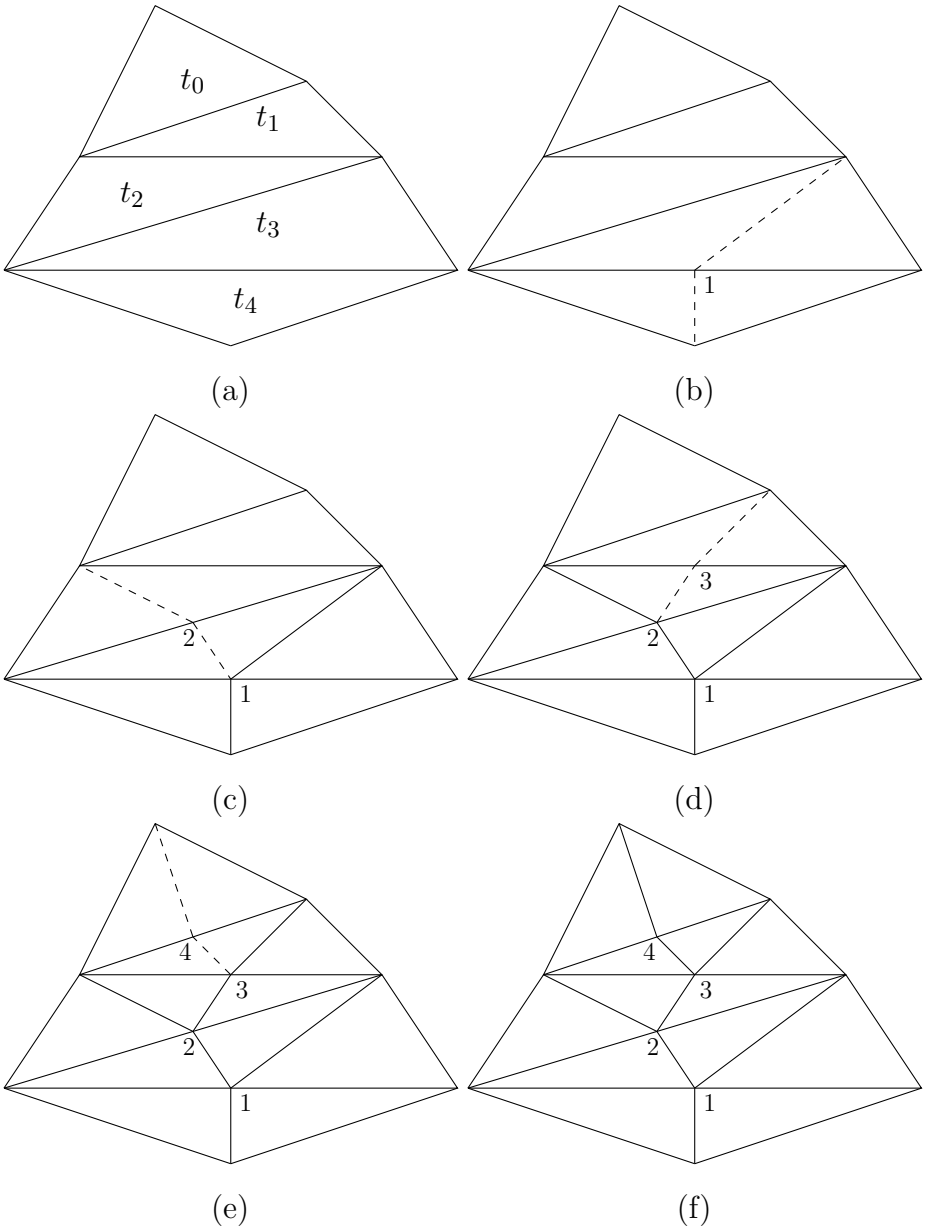


Figura 4.4: Refinamiento LEPP-Bisección del triángulo t_0 [11]

4.1.3.2. LEPP-centroide

El algoritmo LEPP-Centroide utiliza la técnica de la propagación por la arista más larga (*Longest edge propagation path*) donde, se busca romper los triángulos que posean ángulos menores a un cierto valor límite.

Primero, para aplicar este refinamiento es necesario encontrar un triángulo candidato, t, el cual posea un ángulo menor a una cota pre establecida. A continuación se busca la arista terminal, E, a partir del LEPP(t).

Al tener identificada la arista terminal, E, se inserta el centroide del cuadrilátero definido por los triángulos terminales adyacentes a la arista terminal, E.

En nuestro caso utilizamos la técnica de insertar el centroide del cuadrilatero definido por los triángulos terminales. Luego de esto, se aplica de forma recursiva el procedimiento de intercambio de aristas, sobre cada uno de los triángulos nuevos hasta obtener una nueva malla. [4, 13].

Algoritmo 2: LEPP-Centroide(T,α)

Entrada: T: Triángulación inicial para el refinamiento.
Entrada: α : Valor del angulo mínimo deseado.

```
1 mientras Buscar(△n < α)EnLaTriangulacion (α,T) == TRUE hacer
2 |  ti ← SeleccionarTrianguloMalaCalidad (α,T);
3 |  mientras ti existe en T hacer
4 |  |  tn ← LEPP( ti);
5 |  |  punto ← CalcularCentroide( tn, T);
6 |  |  T← EjecutarLeppCentroide( punto, tn, T);
7 |  fin
8 fin
```

En la figura 4.5 se ilustra el refinamiento basado en LEPP-Centroide, del triángulo t_0 sobre la triangularización inicial. Primero en la figura 4.5(a) se busca el Lepp(t_0)= $\{t_0, t_1, t_2, t_3, t_4\}$, luego de esto se busca el centroide de los triángulos terminales, al tener el centroide se elimina la arista terminal y se crean cuatro nuevos triángulos definidos por una arista, la cual nace de cada uno de los vértices (de este nuevo polígono) hasta llegar al centroide, como muestra la figura 2(b), este proceso se repite hasta la bisección del triángulo inicial t_0.

Definición : Para cualquier triangulación T, el uso repetido de la técnica Backward Longest Edge Centroide Improvement sobre los peores triángulos de la malla (con el ángulo más pequeño $\alpha < \theta_{tol}$) produce una triangulación de calidad de ángulos mayores o iguales a θ_{tol} (ángulo de tolerancia mínima) [14].

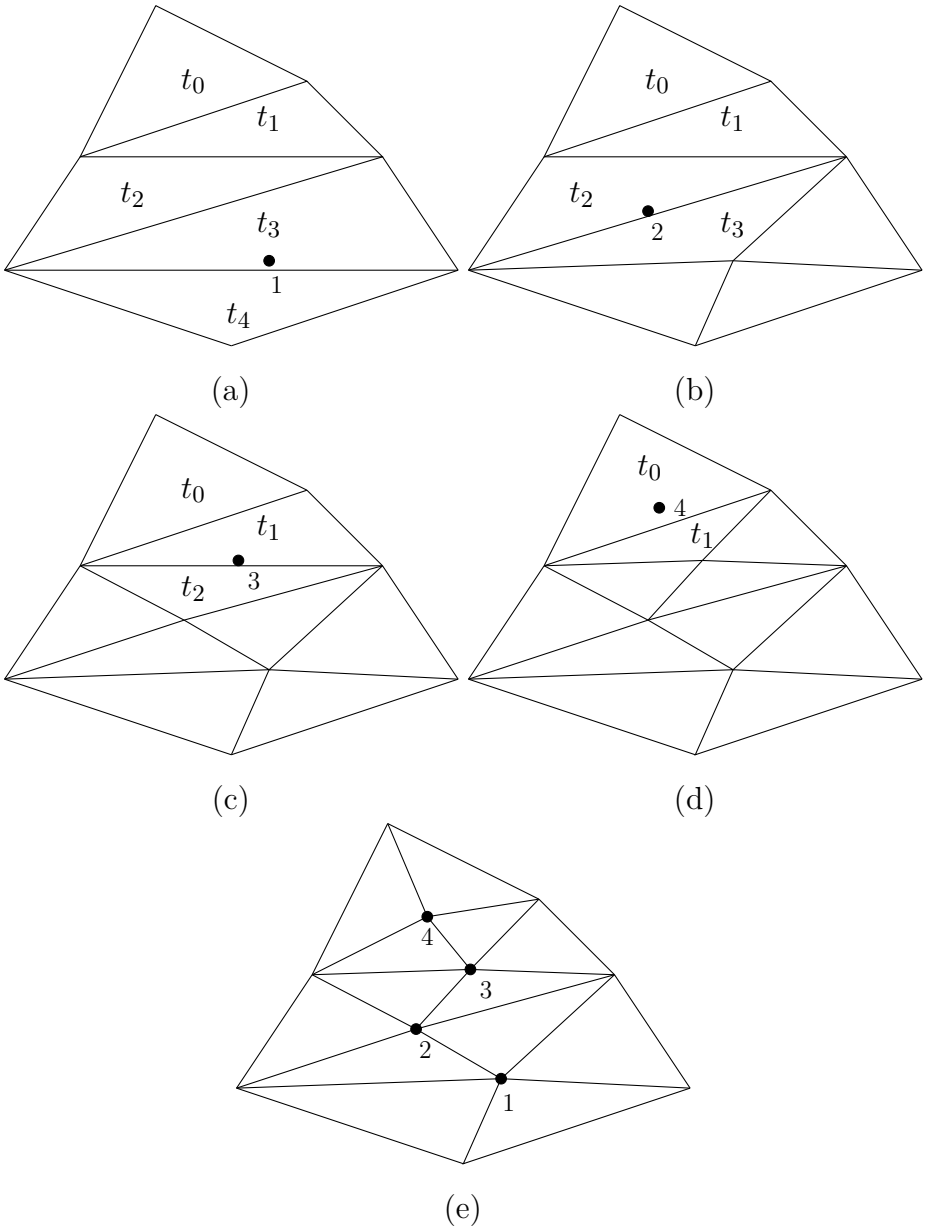


Figura 4.5: Refinamiento LEPP-Centroide del triángulo t_0 [11]

4.1.3.3. LEPP-Delaunay

El algoritmo LEPP-Delaunay utiliza la técnica de la propagación por la arista más larga (*Longest edge propagation path*) donde también se busca romper los triángulos que posean ángulos menores a un cierto valor límite.

Primero, para aplicar este refinamiento es necesario encontrar un triángulo candidato, t, el cual posea un ángulo menor a una cota pre establecida. A continuación se busca la arista terminal, E, a partir del LEPP(t).

Al tener identificada la arista terminal, E, existen dos criterios para el refinamiento:

(a) Insertar el punto medio en la arista terminal.

(b) Insertar el centroide del cuadrilátero definido por los triángulos terminales adyacentes a la arista terminal E.

En nuestro caso utilizamos la técnica de insertar el centroide del cuadrilatero definido por los triángulos terminales. Luego de esto, se aplica de forma recursiva el procedimiento de intercambio de aristas (del algoritmo Delaunay) sobre cada uno de los triángulos nuevos y sobre algunos de sus vecinos hasta obtener una nueva malla Delaunay [4, 13].

Algoritmo 3: LEPP-Delaunay(T,α)

Entrada: T: Triángulación inicial para el refinamiento.
Entrada: α : Valor del angulo mínimo deseado.

```
1 mientras Buscar(△_n < α)EnLaTriangulacion (α,T) == TRUE hacer
2 |   t_i ← SeleccionarTrianguloMalaCalidad (α,T);
3 |   mientras t_i existe en T hacer
4 |   |   t_n ← LEPP( t_i);
5 |   |   punto ← CalcularCentroide( t_n, T);
6 |   |   T← InsercionDelaunay( punto, t_n, T);
7 |   fin
8 fin
```

En la figura 4.6 se ilustra el refinamiento basado en LEPP-Delaunay-Centroide, del triángulo t_0 sobre la triangularización inicial. Primero en la figura 4.6(a) se busca el LEPP(t_0)= $\{t_0, t_1, t_2, t_3, t_4\}$, luego de esto se busca el centroide de los triángulos terminales, al tener el centroide se elimina la arista terminal y se crean cuatro nuevos triángulos definidos por una arista, la cual nace de cada uno de los vértices (de este nuevo polígono) hasta llegar al centroide, luego de eso se ejecuta la inserción Delaunay, como muestra la figura 3(c) y 3(d), este proceso se repite hasta la bisección del triángulo inicial t_0.

Definición : Para cualquier triangulación T, el uso repetido de la técnica Backward Longet Edge Delaunay Improvement sobre los peores triángulos de la malla (con el ángulo más pequeño $\alpha < \theta_{tol}$) produce una triangulación de calidad de ángulos mayores o iguales a θ_{tol} (ángulo de tolerancia mínima) [14].

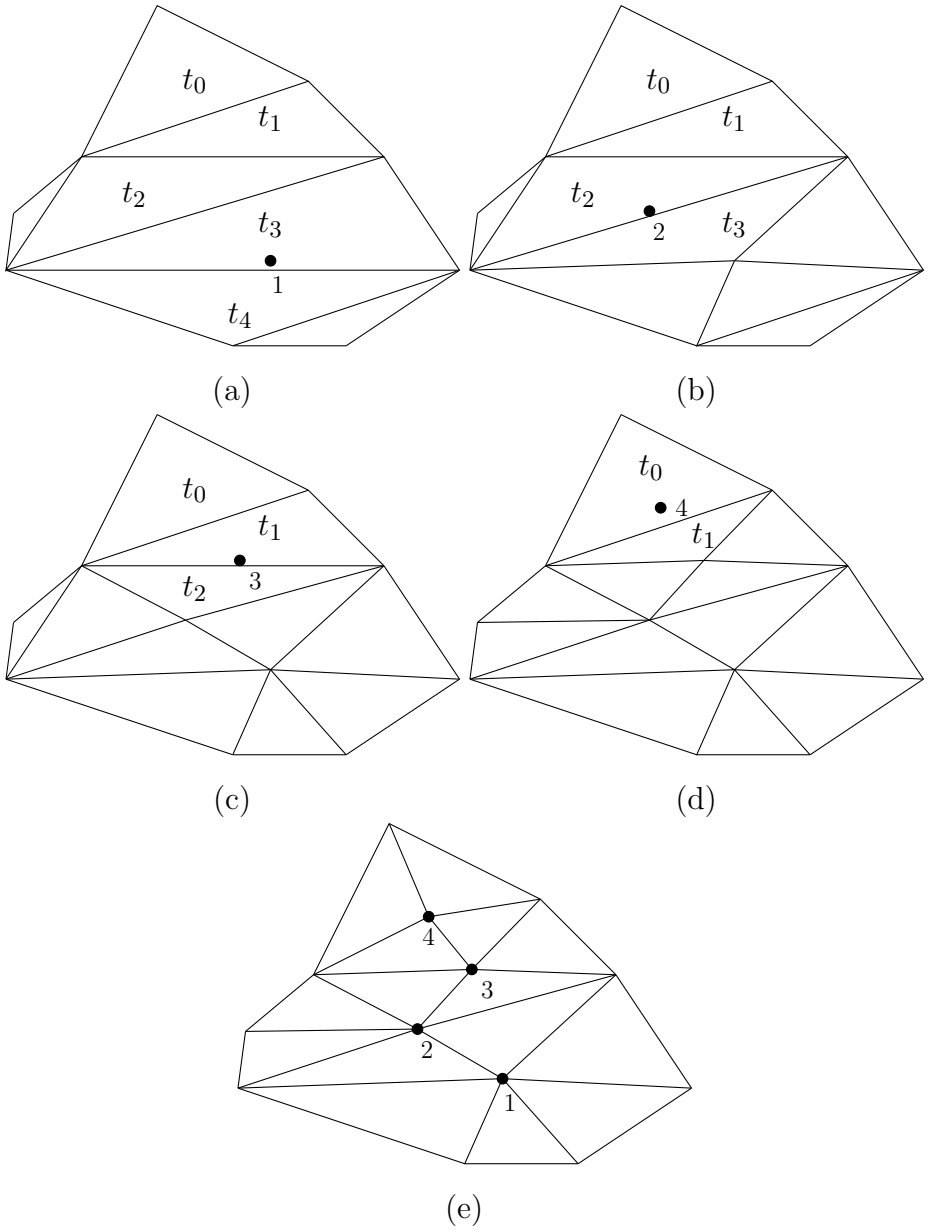


Figura 4.6: Refinamiento LEPP-Delaunay del triángulo t_0 [11]

Capítulo 5

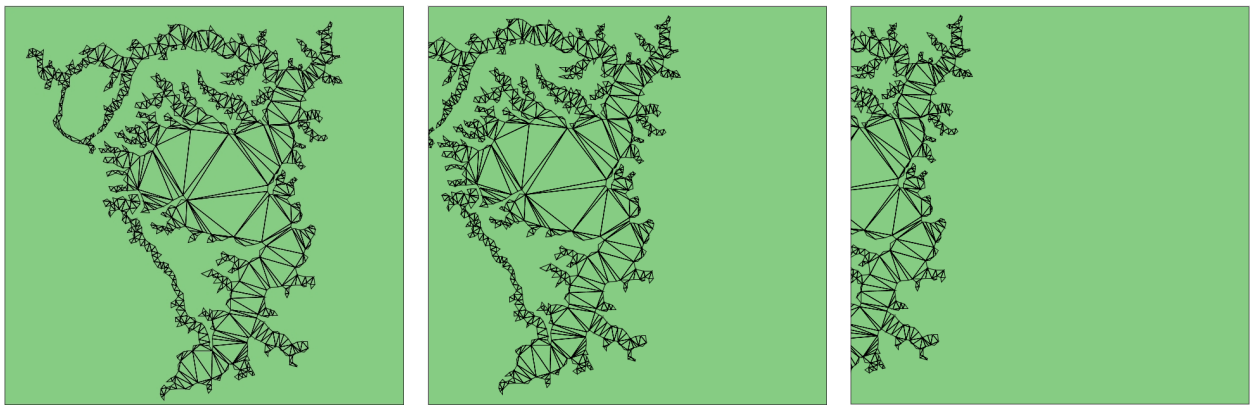
Pruebas de funcionalidad

5.1. Pruebas de gráficos

En esta sección se aborda las pruebas asociadas a la funcionalidad de software, su comportamiento con el puntero y la sobrecarga de figuras en la gráfica.

5.1.1. Movilidad

Corresponde a la interacción entre el puntero y la zona de dibujo canvas, en la cual se obtiene la coordenada x,y asociada a la dimension de altura y anchura del canvas, luego de esto, se aplica una transformación lineal, entre la zona limitada del canvas y la capa de dibujo, asociada a la malla. Con las coordenadas $x´$,$y´$ transformadas, estas se concatenan a la malla lo cual produce una traslado de la malla al lugar donde se encuentra el puntero.



(a) (b) (c)

Figura 5.1: En la imagen (a) se presiona el puntero, luego en la figura (b) se desplaza el puntero hacia la izquierda de la zona de dibujo canvas, por último en la figura (c) se libera el desplazamiento del puntero.

5.1.2. Acercamiento

Corresponde a la interacción entre la rueda del puntero y la zona de dibujo canvas, en la cual se obtiene la coordenada x,y asociada a la dimensión de altura y anchura del canvas, luego de esto, se aplica una transformación lineal de escalado y de traslación, entre la zona limitada del canvas y la capa de dibujo, asociada a la malla. Con las coordenadas $x´$,$y´$ transformadas, estas se concatenan a la malla lo cual produce un traslado y escalado de la malla al lugar donde se encuentra el puntero.

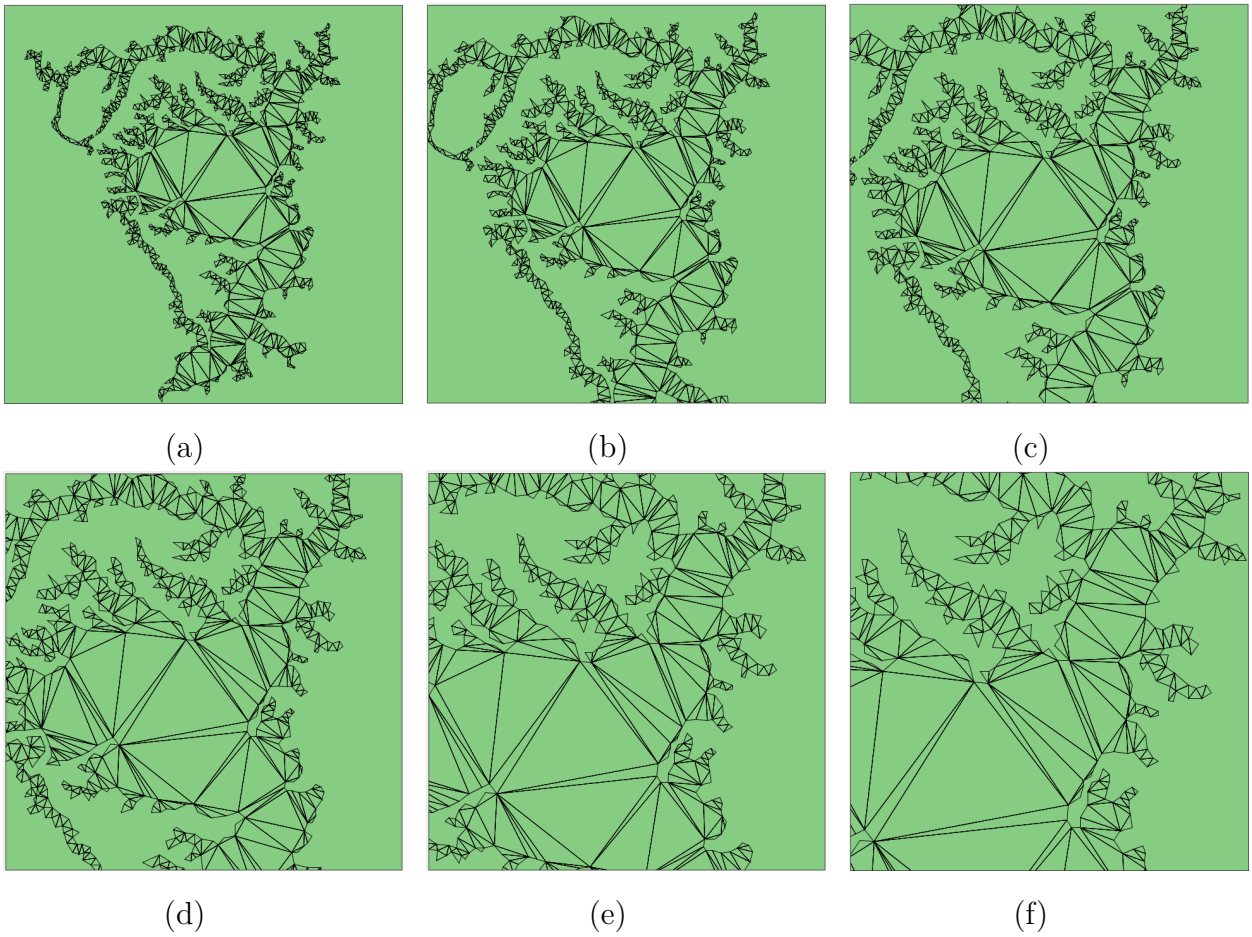


Figura 5.2: En la imagen (a) se posiciona el puntero en la localización donde se quiere hacer el acercamiento a la figura, luego en la figura (b) se gira la rueda del puntero hacia el lugar seleccionado de la zona de dibujo canvas, y asi sucesivamente hasta la figura (f).

5.1.3. Sobrecarga de figuras

Corresponde a la funcionalidad del panel log, en el cual se mide los fps[1], aps[2] y la cantidad de figuras pintadas en la zona de dibujo, también hay que tener en cuenta las características del computador utilizado. Al sobrecargar con figuras en la zona de dibujo, baja la cantidad de fps[1] y aumenta la cantidad de aps[2] para que se pueda ver la figura en la zona de dibujo.

1. En la figura 5.3(a) se carga una malla de 1180 figuras triangulares, el motor gráfico dibuja a 50 fps[1] y actualiza a 61 aps[2], promedio.

2. En la figura 5.3(b) se carga una malla de 15926 figuras triangulares, el motor gráfico dibuja a 7 fps[1] y actualiza a 66 aps[2], promedio.

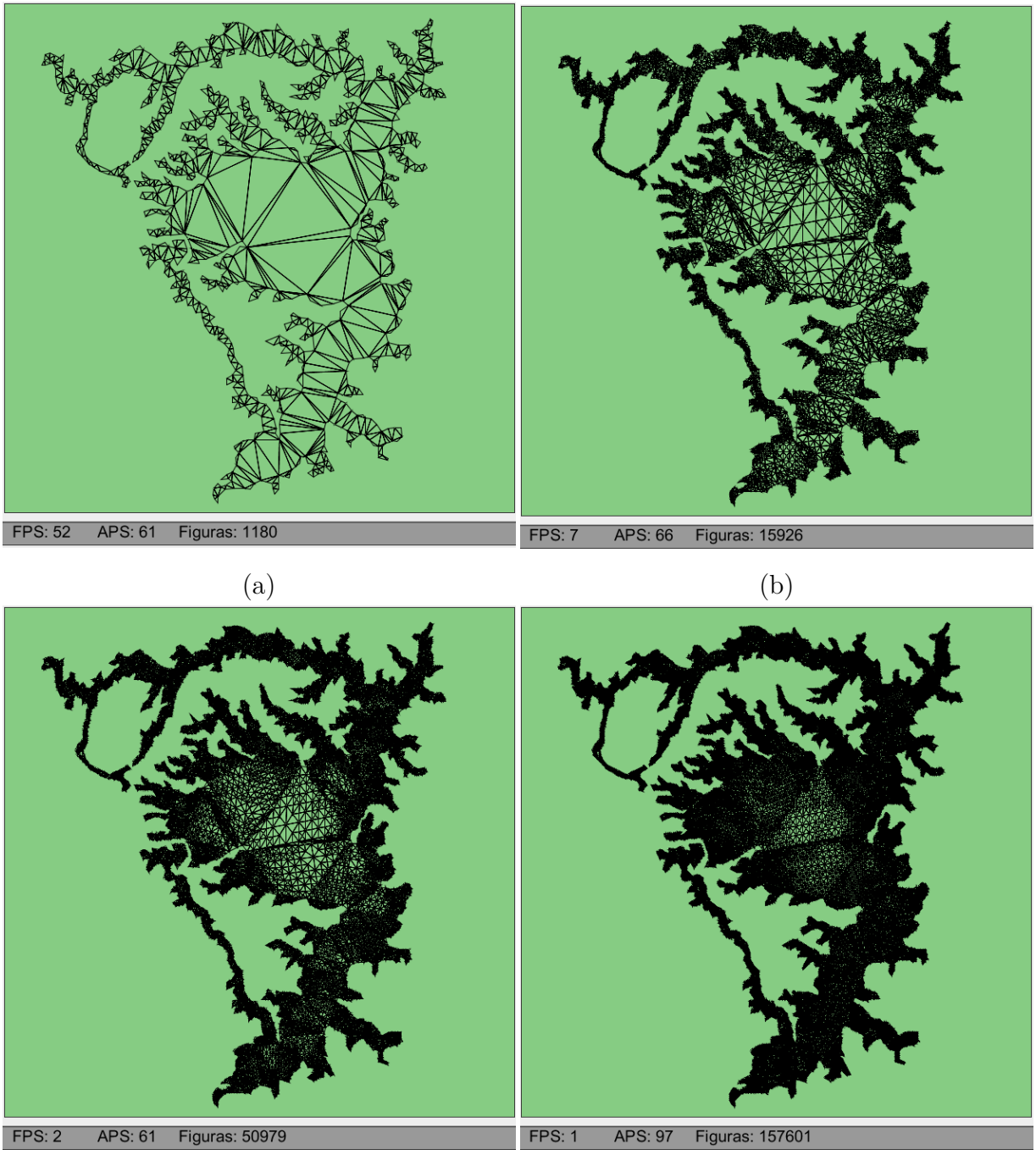
3. En la figura 5.3(c) se carga una malla de 50979 figuras triangulares, el motor gráfico dibuja a 2 fps[1] y actualiza a 61 aps[2], promedio.

4. En la figura 5.3(d) se carga una malla de 157601 figuras triangulares, el motor gráfico dibuja a 1 fps[1] y actualiza a 97 aps[2], promedio.

Al duplicar la cantidad de figuras triangulares en la figura 5.3(d) el motor gráfico se queda sin memoria para guardar la información de la cantidad de datos.

[1]Los FPS representan cuadros por segundo, una medida de cuántas imágenes consecutivas únicas puede manejar una cámara por segundo

[2]Un sistema de planificación y programación avanzada (APS) es un tipo de sistema de información para admitir la planificación o la programación, y generalmente tienen interfaces gráficas de usuario



(a) (b)

(c) (d)

Figura 5.3: Sobrecarga de figuras.

5.2. Pruebas de selección

En esta sección se muestra la funcionalidad de los tipos de selección en la malla de figuras.

5.2.1. Selección de un triángulo

Al seleccionar un triangulo se muestra en el panel de la derecha las características de de esa figura. El identificador de la figura seleccionada, El identificador de los tres vértices y sus coordenadas. Las tres aristas que componen la figura y la distancia de ellas. También los identificadores de los vecinos y los triángulos que los componen.

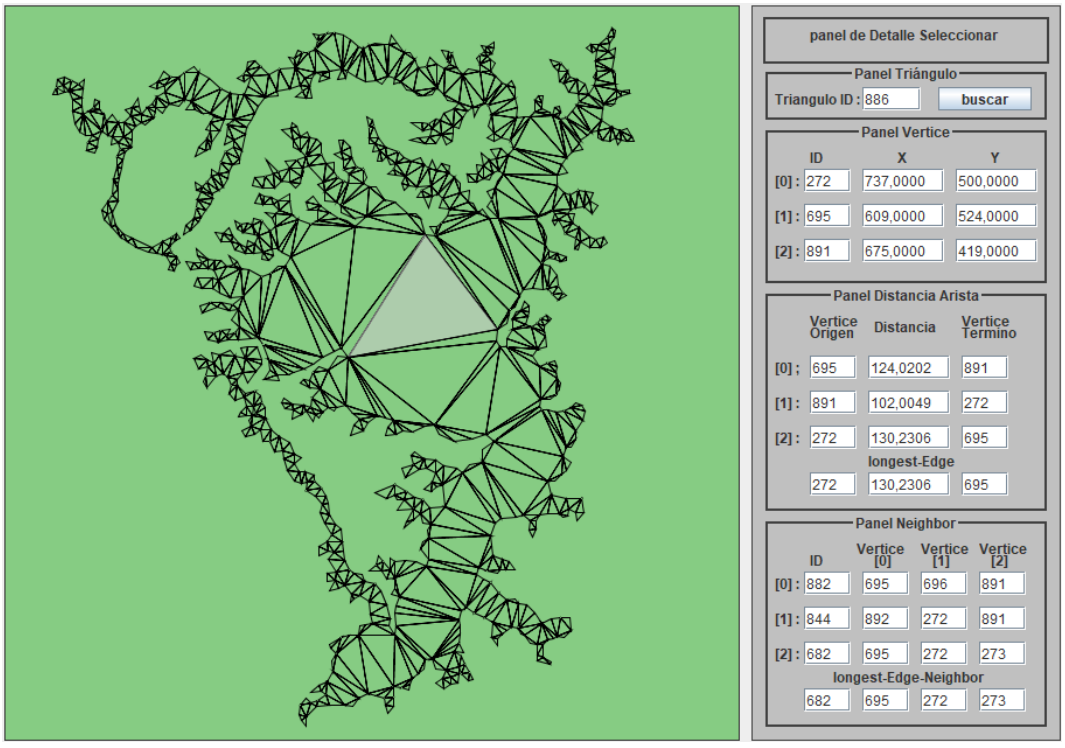


Figura 5.4: Selección de un triángulo.

5.2.2. Selección por circunsferencia

Al ocupar el tipo de selección circular, se toman todos los vértices de los triángulos contenidos en la circunsferencia y estos son pintados de gris.

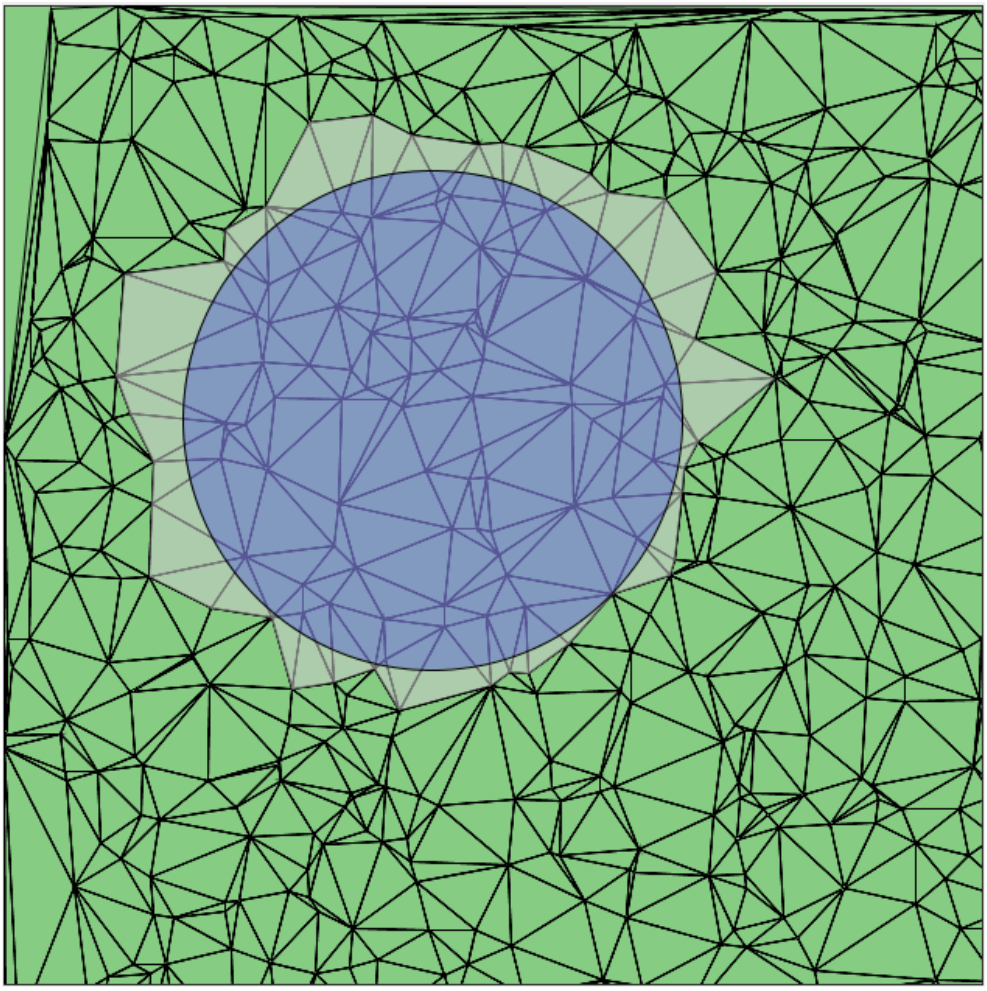


Figura 5.5: Selección de una circunsferencia.

5.2.3. Selección por distancias

En esta sección se selecciona todos los triángulos cuyas aristas cumplan con la condición, que la arista más larga sea menor al valor ingresado en el panel de la derecha.

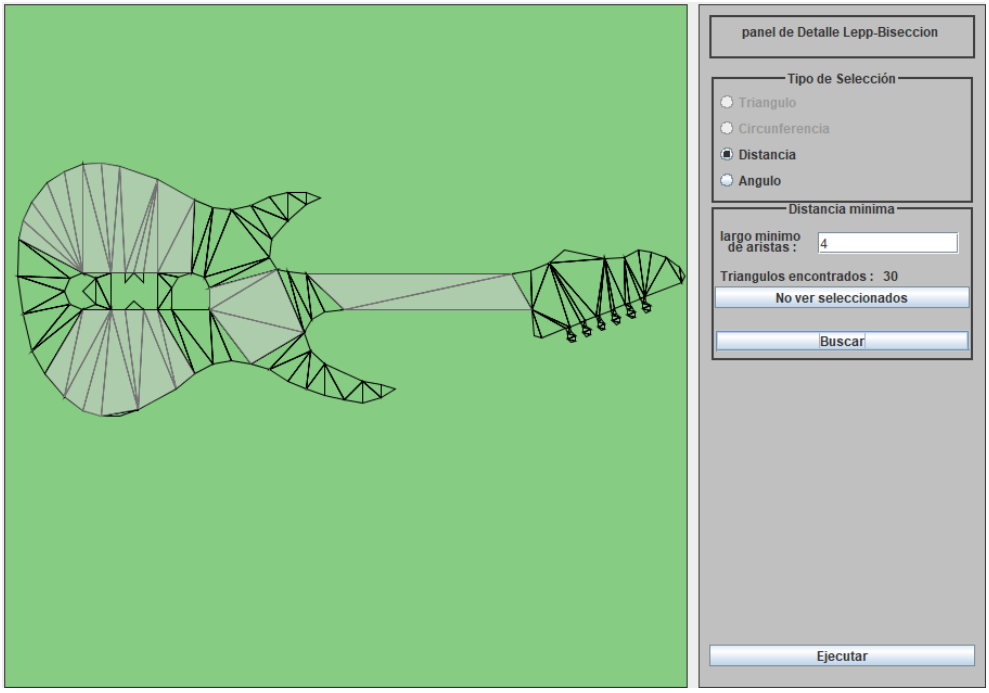


Figura 5.6: Selección de triángulos por distancia.

5.2.4. Selección por ángulos

En esta sección se selecciona todos los triángulos, que cumplan con la condición del ángulo más pequeño, que sea menor al valor ingresado en el panel de la derecha.

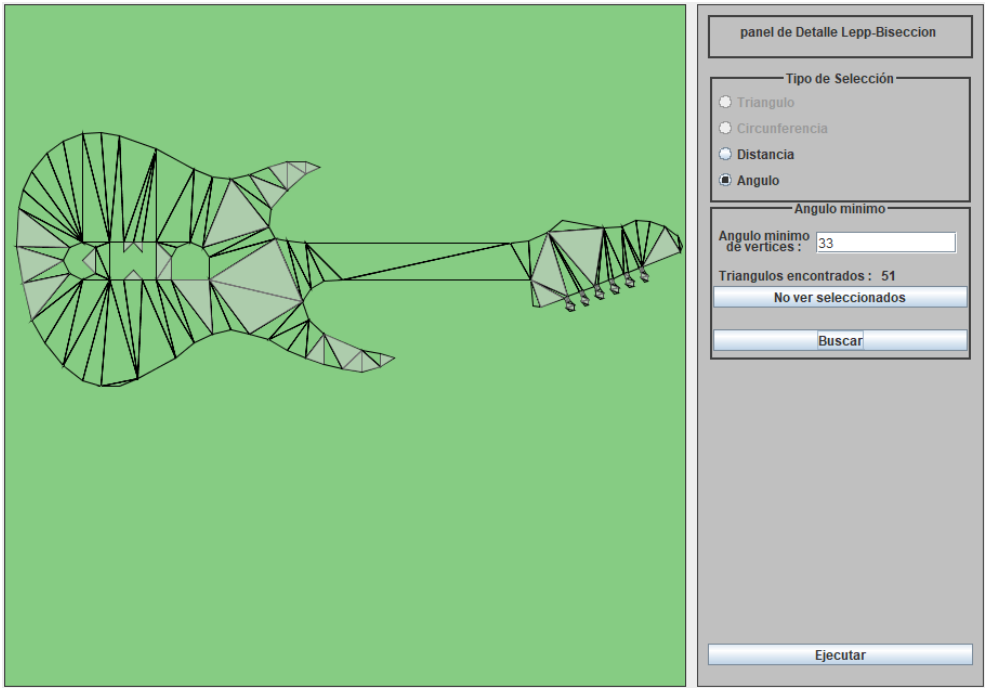


Figura 5.7: Selección de triángulos por angulo.

5.3. Pruebas de algoritmos

En esta sección se muestra la funcionalidad de los algoritmos dependiendo del tipo de selección.

5.3.1. LEPP

Se utiliza el algoritmo (*Longest edge propagation path*) para visualizar su funcionalidad. Se muestra el triángulo seleccionado para el LEEP en gris oscuro y su área de propagación en gris.

5.3.1.1. Triángulo

En esta sección se selecciona un triángulo y se ejecuta el algoritmo LEPP.

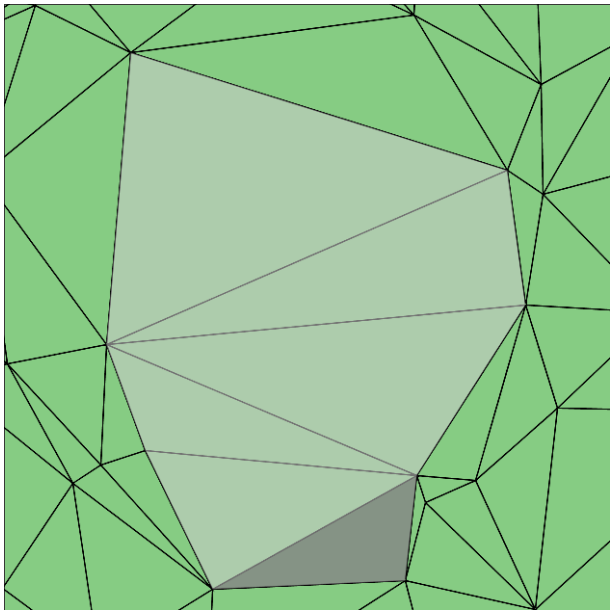
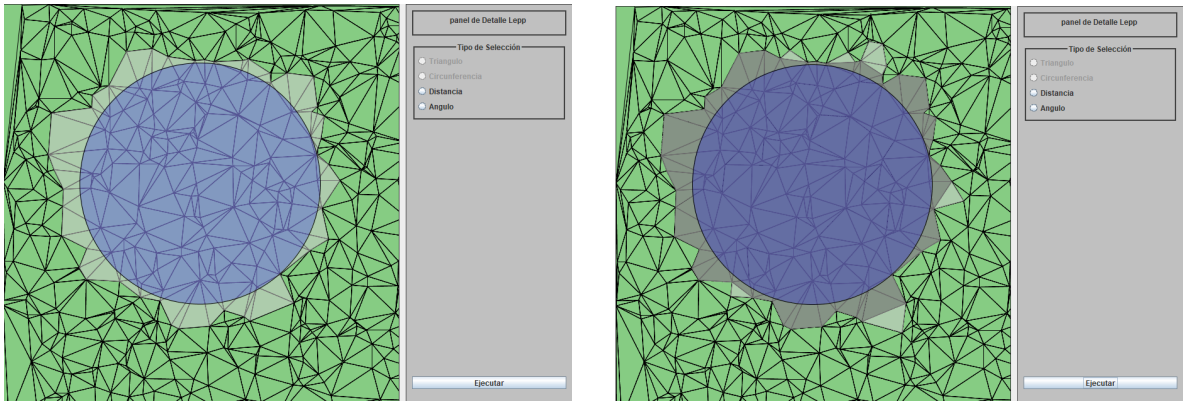


Figura 5.8: LEPP de un triángulo.

5.3.1.2. Circunsferencia

En esta sección se selecciona un área delimitada por una circunsferencia y todos los triángulos cuyos vértices estén contenidos en la circunsferencia, estos son seleccionados pintando su área de color gris, luego de esto se aplica el algoritmo LEPP y todos aquellos triángulos contenidos son pintados dentro de su LEPP, donde se muestra los triángulos contenidos en la circunsferencia en gris oscuro y su área de propagación en gris.



(a) (b)

Figura 5.9: LEPP en una circunferencia.

5.3.1.3. Distancia

En esta sección se hace una búsqueda en la malla de todos aquellos triángulos cuya arista más larga sea menor al valor ingresado por el usuario.

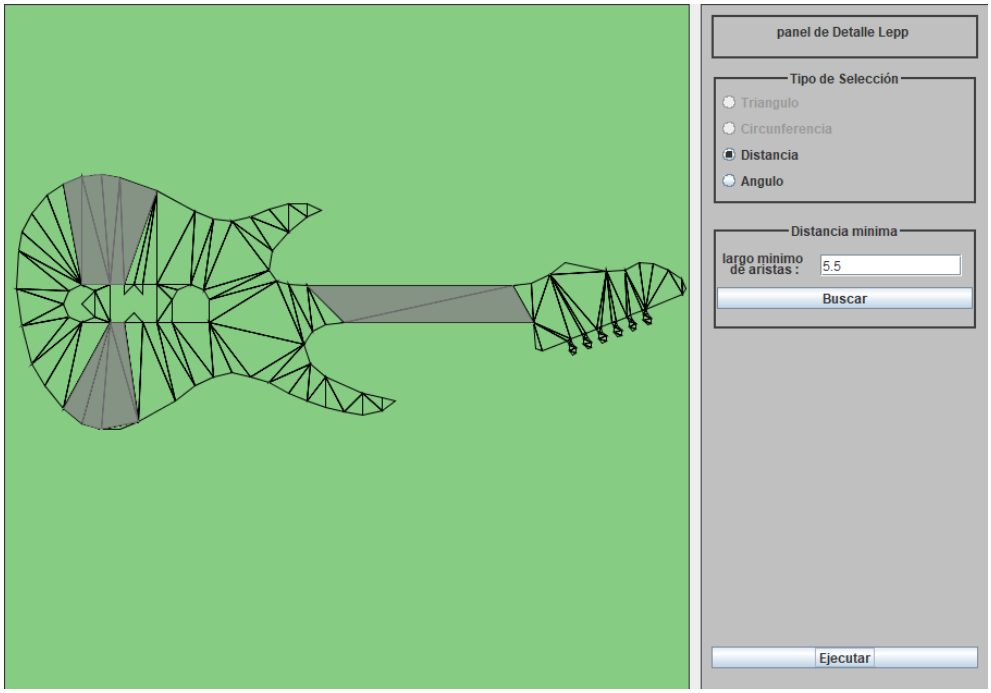


Figura 5.10: Selección de triángulos por la longitud de arista mas larga.

5.3.1.4. Ángulo

En esta sección se hace una búsqueda en la malla de todos aquellos triángulos cuyo ángulo menor, sea menor al valor ingresado por el usuario.

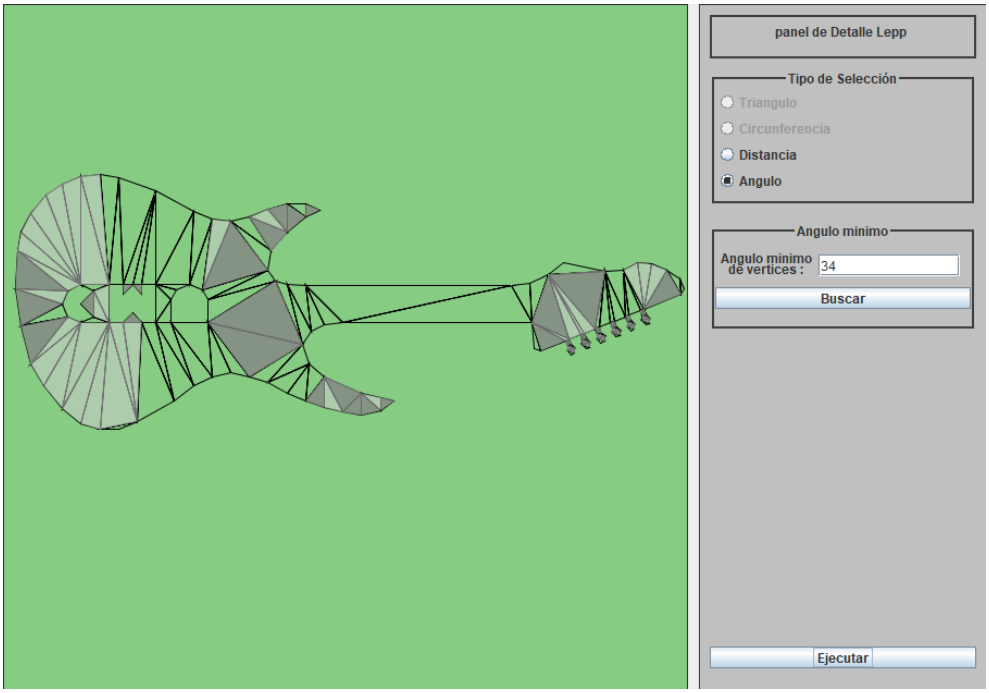


Figura 5.11: Selección de triángulos por ángulo.

5.3.2. LEPP Bisección

En esta sección se utiliza el algoritmo de LEPP Bisección dependiendo del tipo de selección.

5.3.2.1. Triángulo

Para refinar un triángulo seleccionado desde la malla se utiliza el algoritmo de LEPP Bisección.

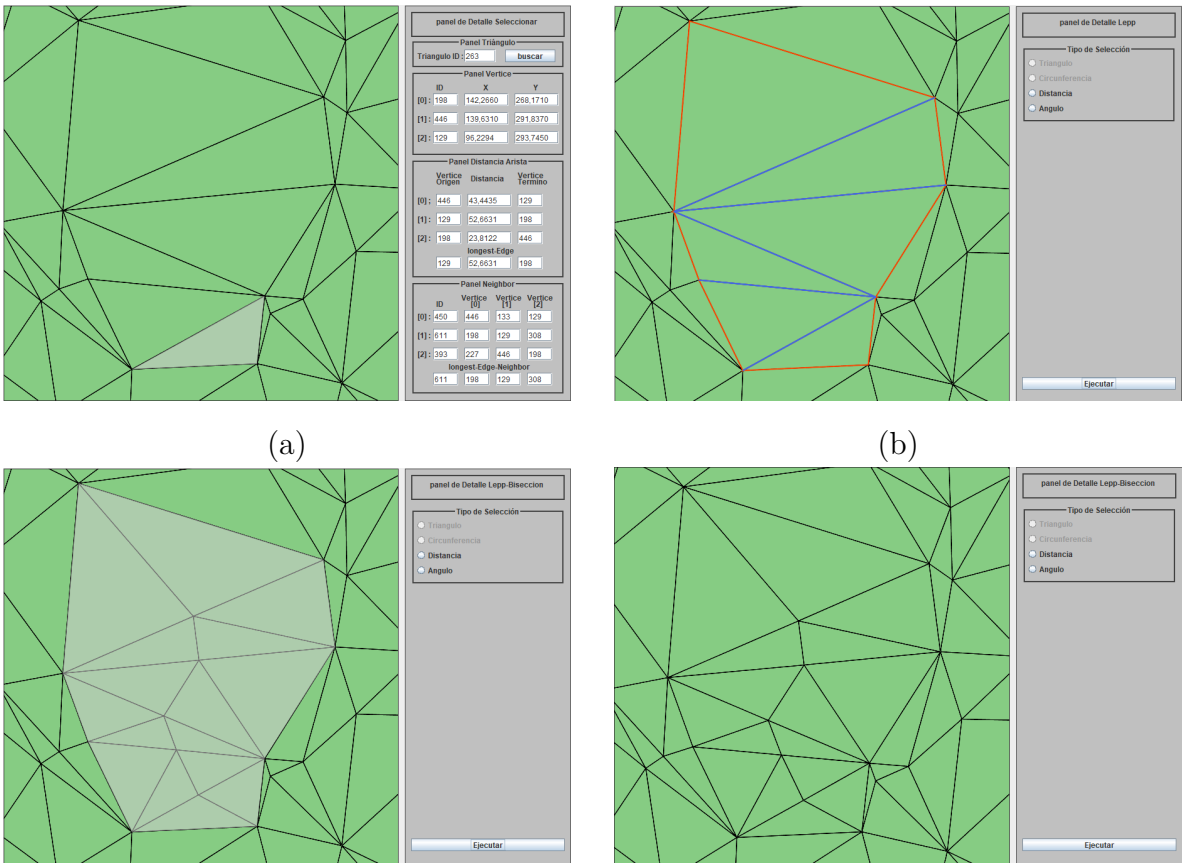


Figura 5.12: LEPP-Bisección de un triángulo.

5.3.2.2. Circunsferencia

Se utiliza el algoritmo LEPP Bisección para refinar los triángulos seleccionados dentro de un área delimitada por la circunsferencia.

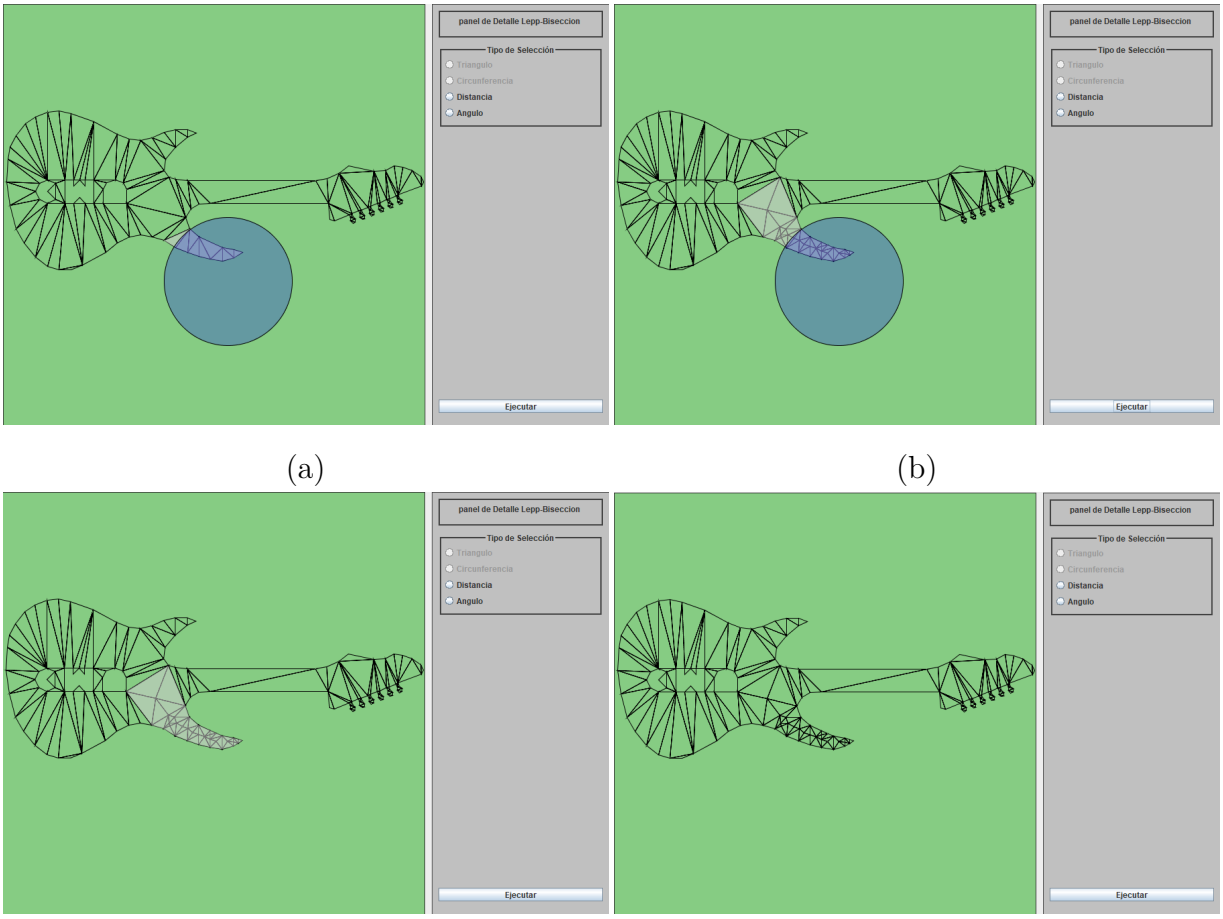


Figura 5.13: Aplicación del algoritmo LEPP-Biseccion para refinar los triángulos dentro del área circular.

5.3.2.3. Distancia

Se buscan todos aquellos triángulos cuyas aristas más largas sean mayores al valor ingresado por el usuario.

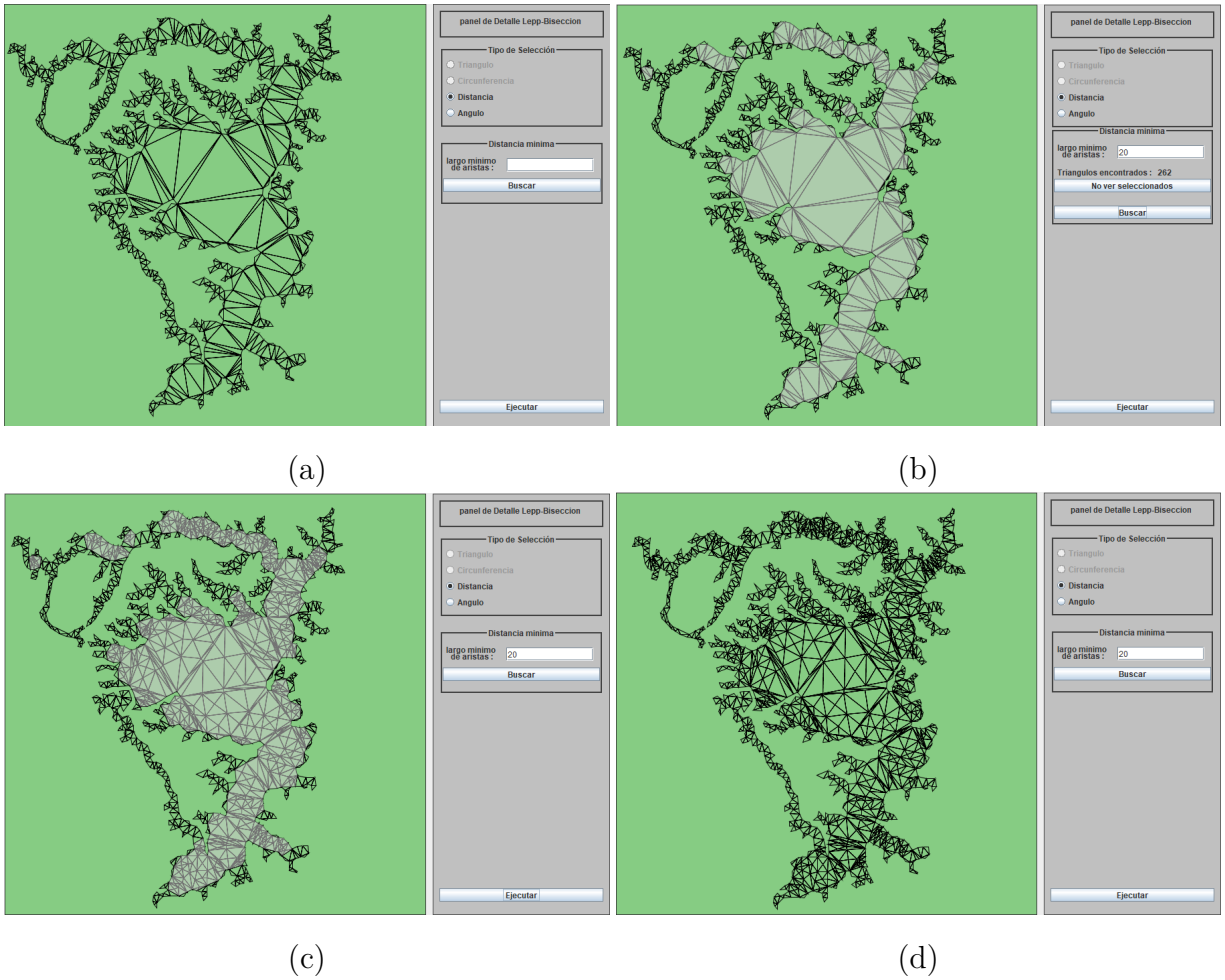


Figura 5.14: Aplicación del algoritmo LEPP-Biseccion para refinar los triángulos seleccionados por longitud de arista más larga.

5.3.3. LEPP-Centroide

En esta sección se utiliza el algoritmo de LEPP-Centroide dependiendo del tipo de selección.

5.3.3.1. Triángulo

Al seleccionar un triángulo en la malla se utiliza el algoritmo de LEPP-Centroide para refinarlo.

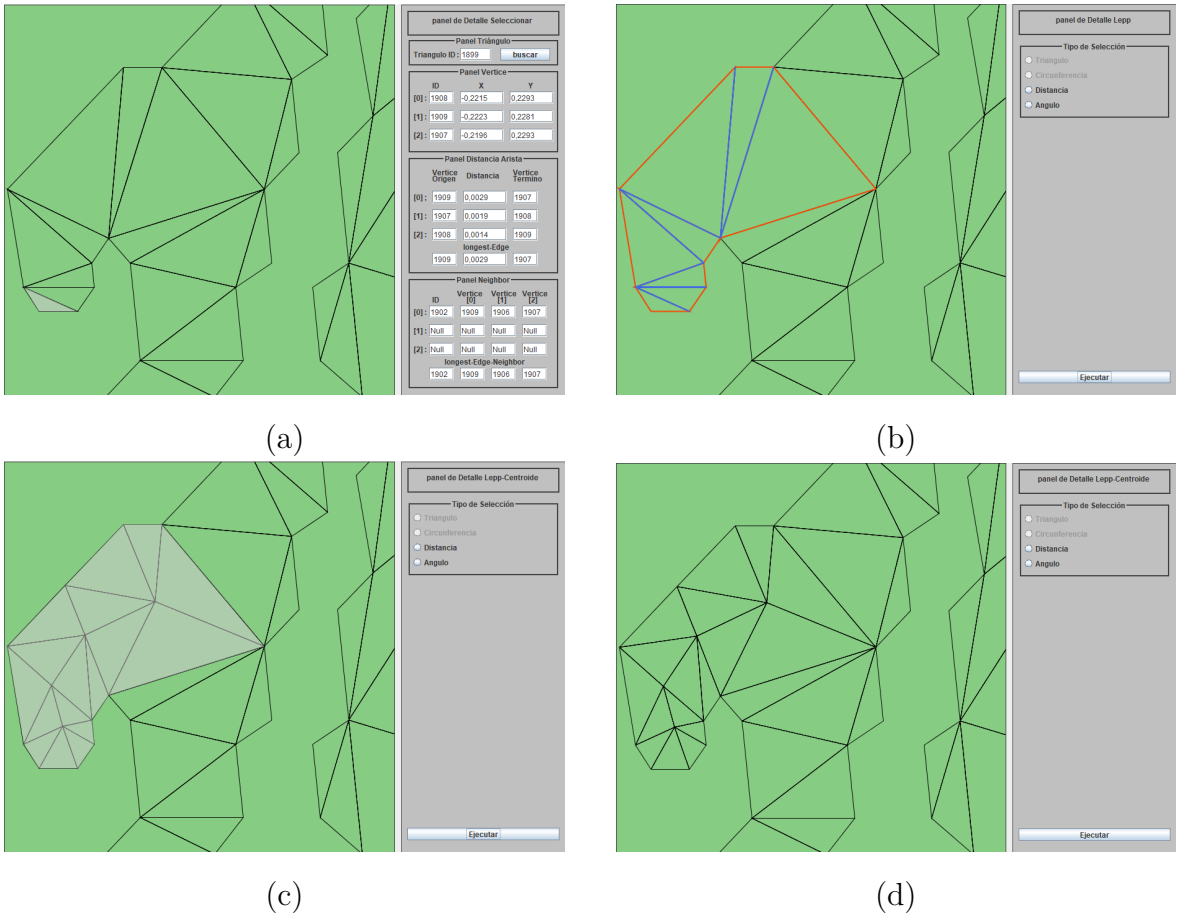


Figura 5.15: Aplicación del algoritmo LEPP-Centroide para refinar un triángulo.

5.3.3.2. Circunsferencia

Se utiliza el algoritmo LEPP-Centroide para refinar los triángulos que están dentro de un área delimitada por la circunsferencia.

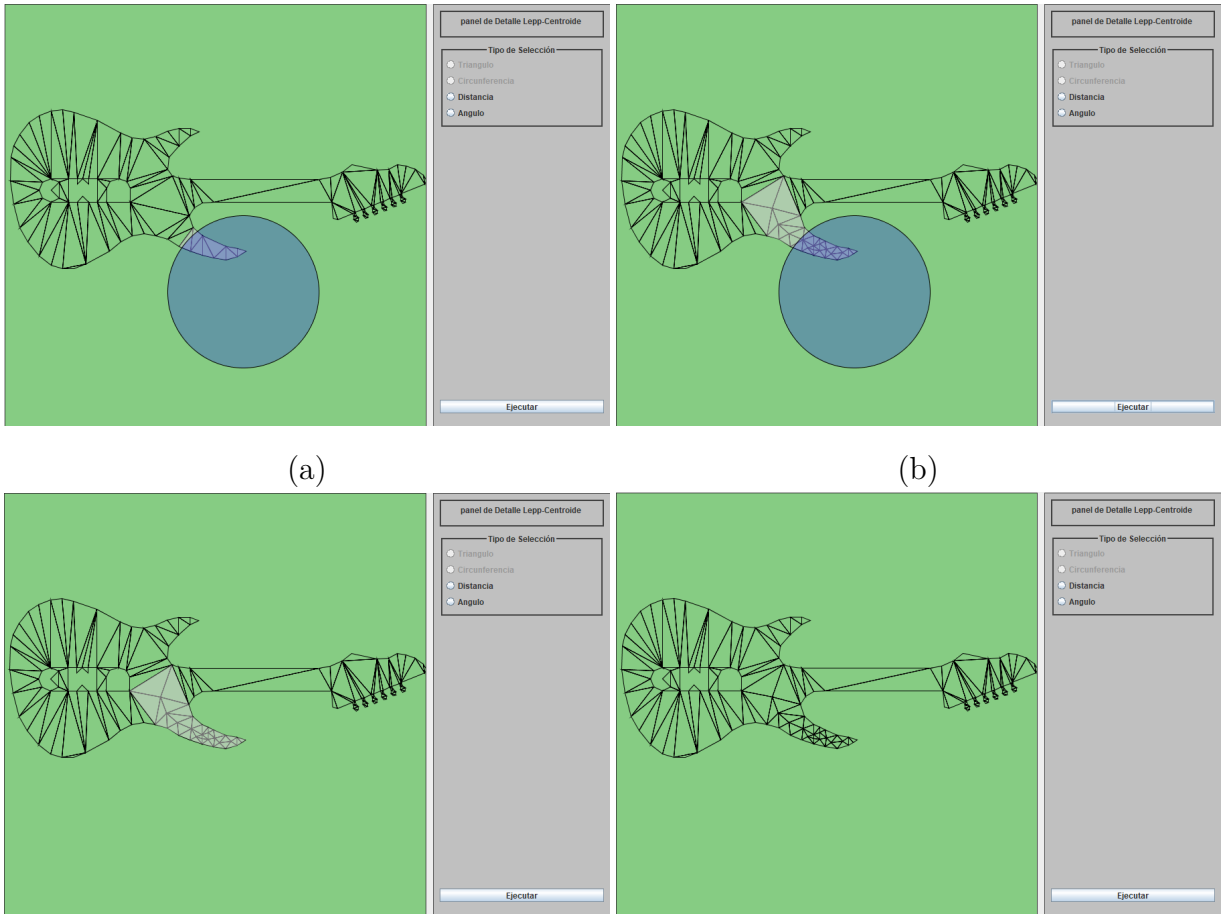


Figura 5.16: LEPP-Centroide de una circunsferencia.

5.3.3.3. Distancia

Se buscan todos aquellos triángulos cuyas aristas más largas sean menores al valor ingresado por el usuario.

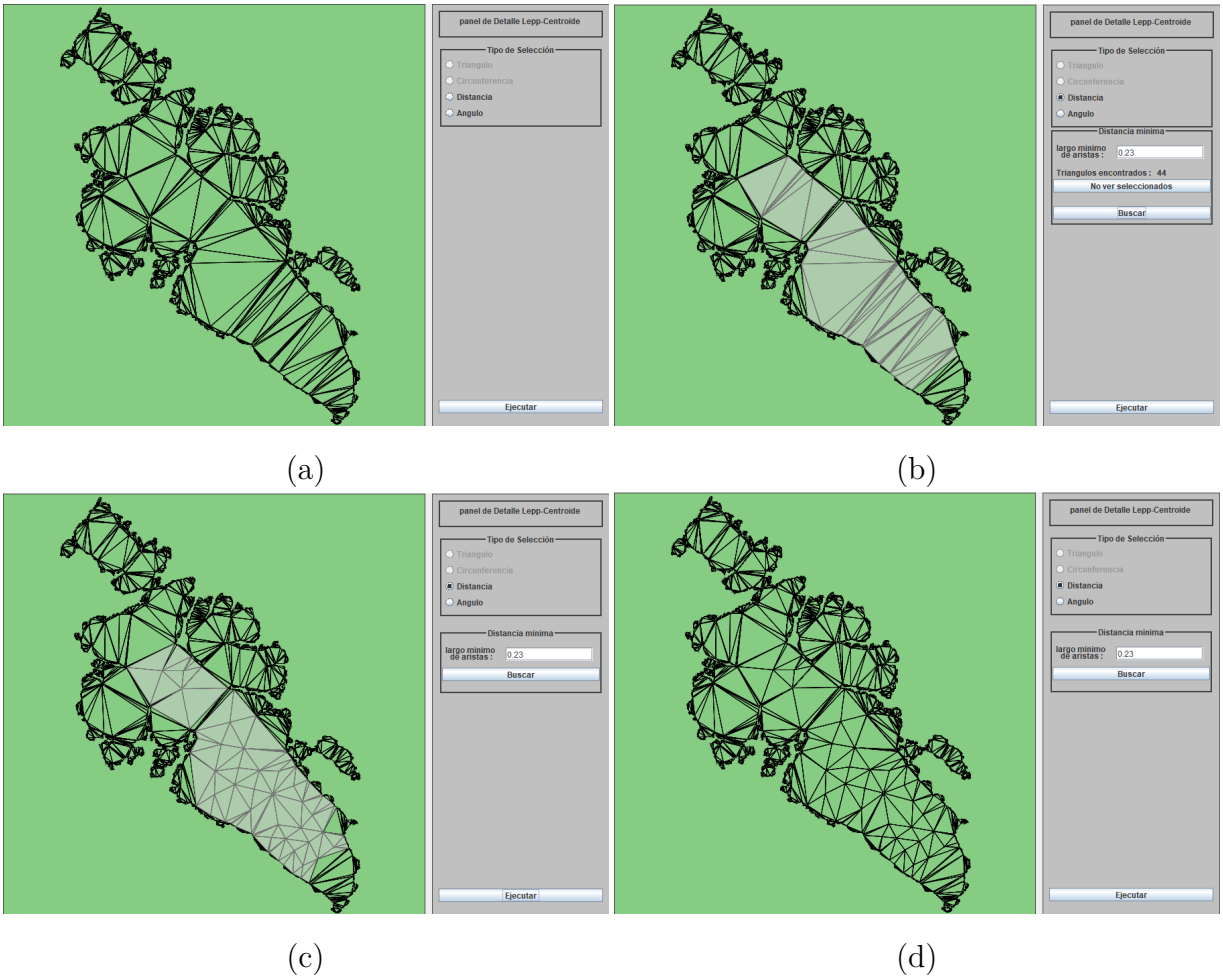


Figura 5.17: Aplicación del algoritmo LEPP-Centroide para refinar triángulos seleccionados.

5.3.3.4. Ángulo

Se buscan todos aquellos triángulos cuyos ángulos sean menores al valor ingresado por el usuario.

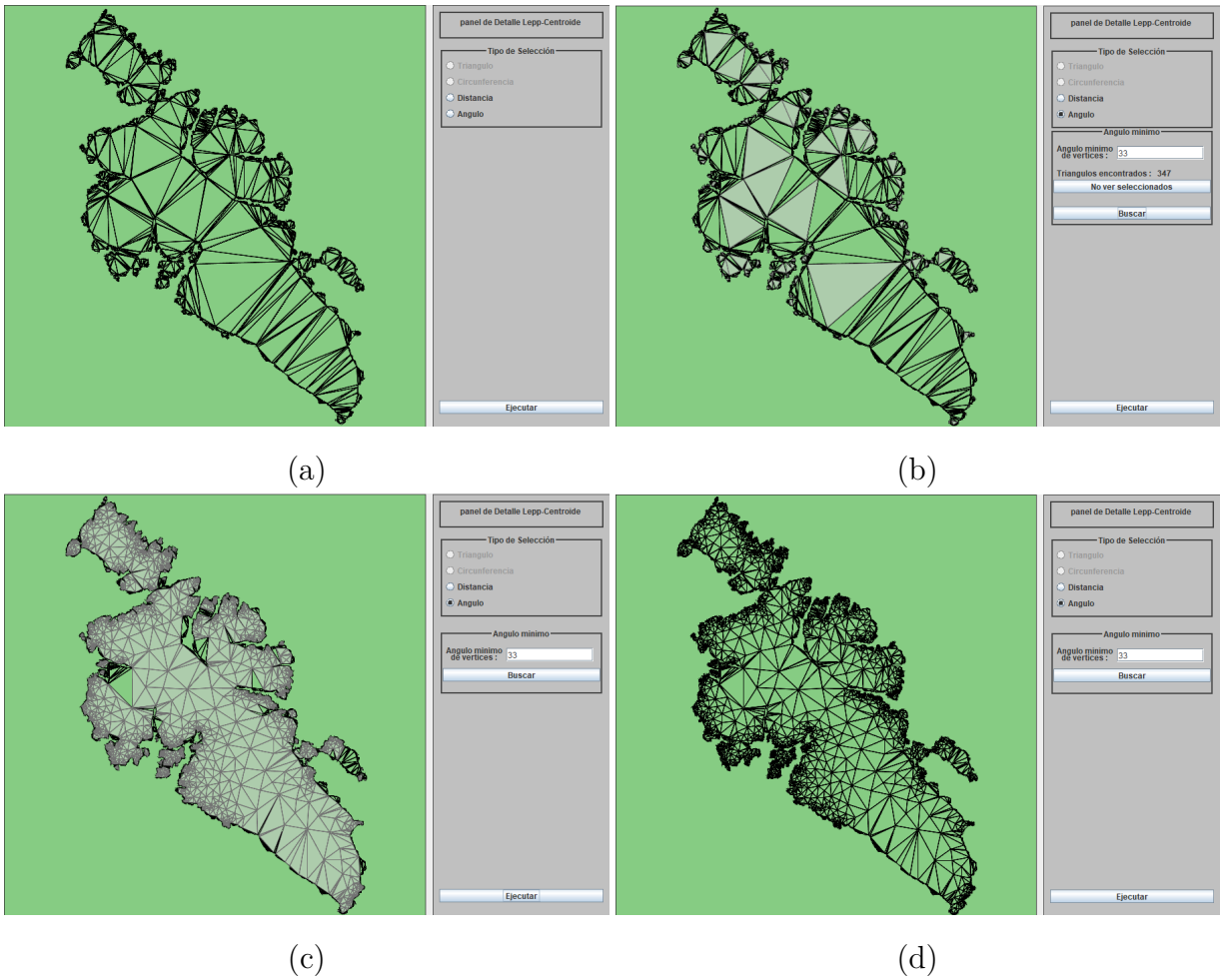


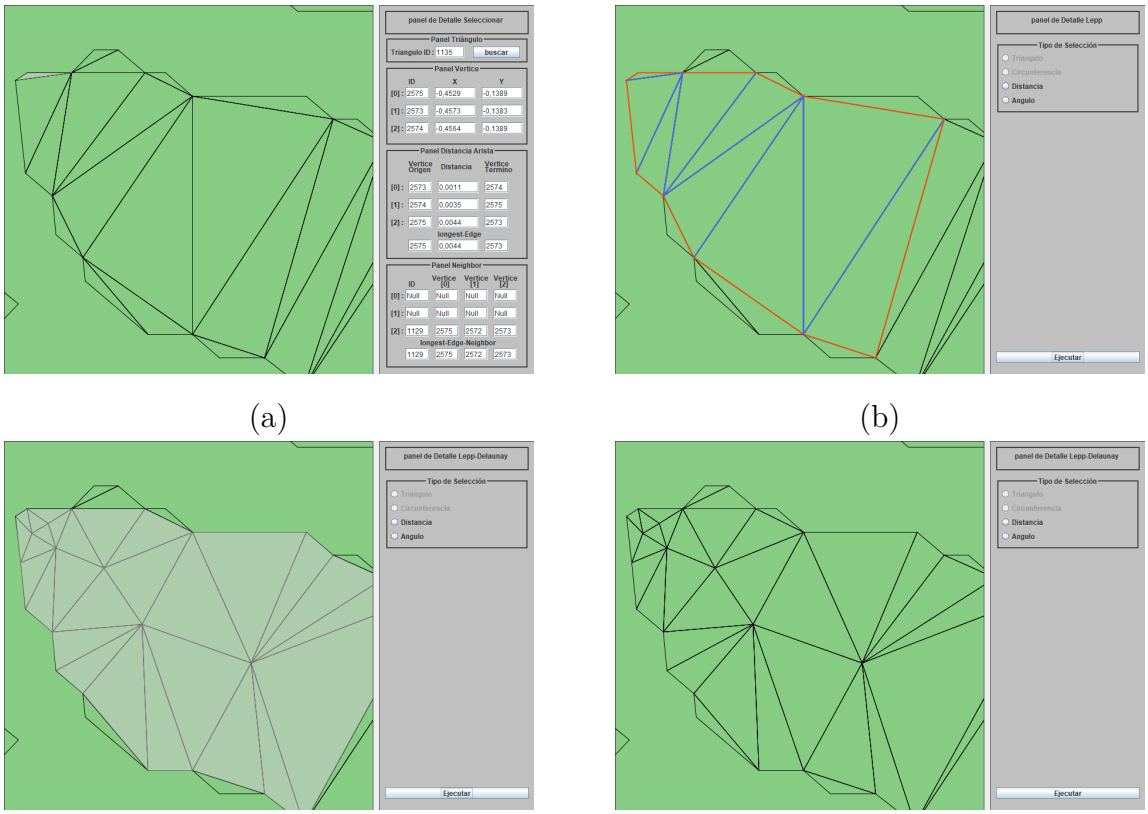
Figura 5.18: Aplicación del algoritmo LEPP-Centroide para refinar los triángulos seleccionados.

5.3.4. LEPP-Delaunay

En esta sección se utiliza el algoritmo de LEPP-Delaunay dependiendo del tipo de selección.

5.3.4.1. Triángulo

Al seleccionar un triángulo en la malla se utiliza el algoritmo de LEPP-Delaunay para refinarlo.



(a) (b)

(c) (d)

Figura 5.19: Aplicación del algoritmo LEPP-Delaunay para refinar los triángulos seleccionados.

5.3.4.2. Circunsferencia

Se utiliza el algoritmo LEPP Delaunay para refinar los triángulos que se encuentran dentro de un área delimitada por la circunsferencia.

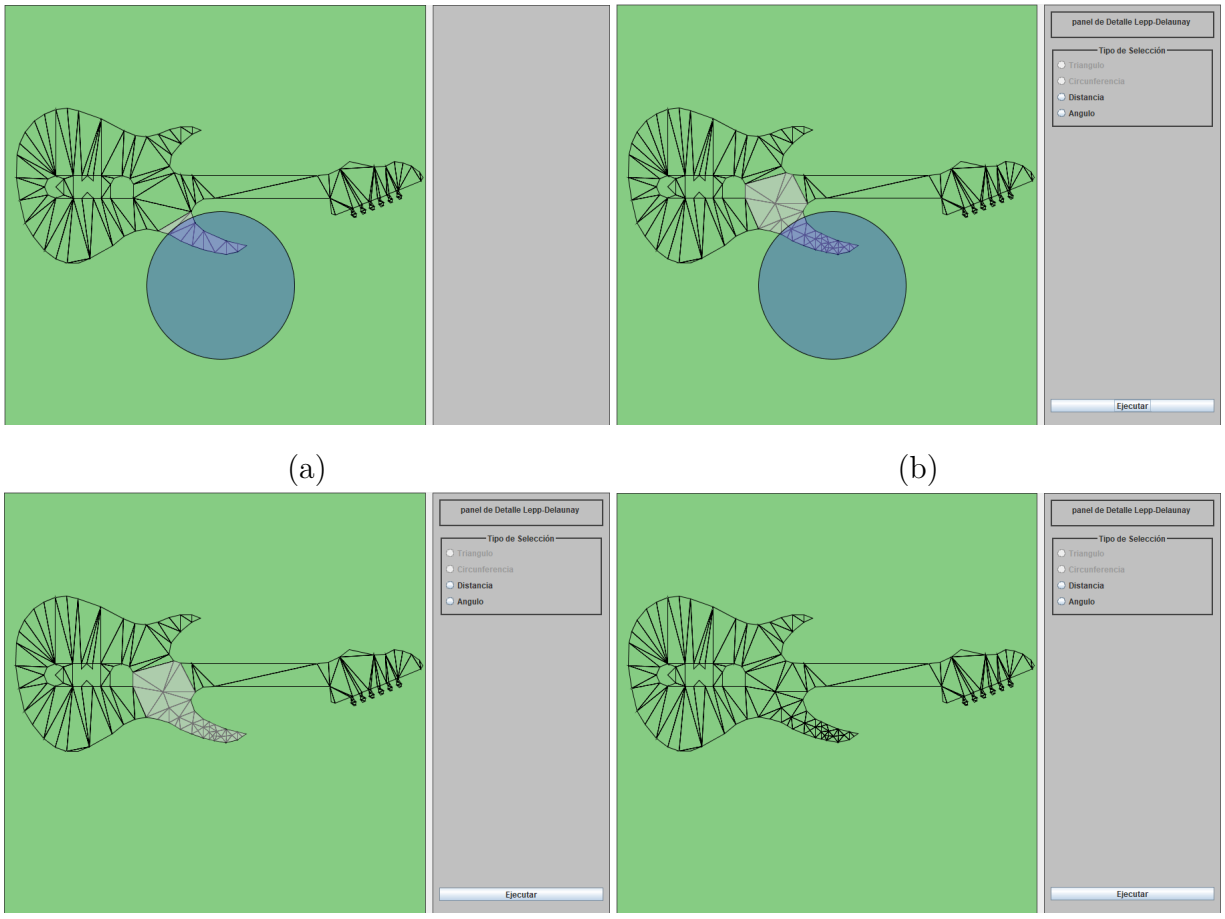


Figura 5.20: Aplicación del algoritmo LEPP-Delaunay para refinar los triángulos dentro de un área circular.

5.3.4.3. Ángulo

Se buscan todos aquellos triángulos cuyos ángulos sean menores al valor ingresado por el usuario.

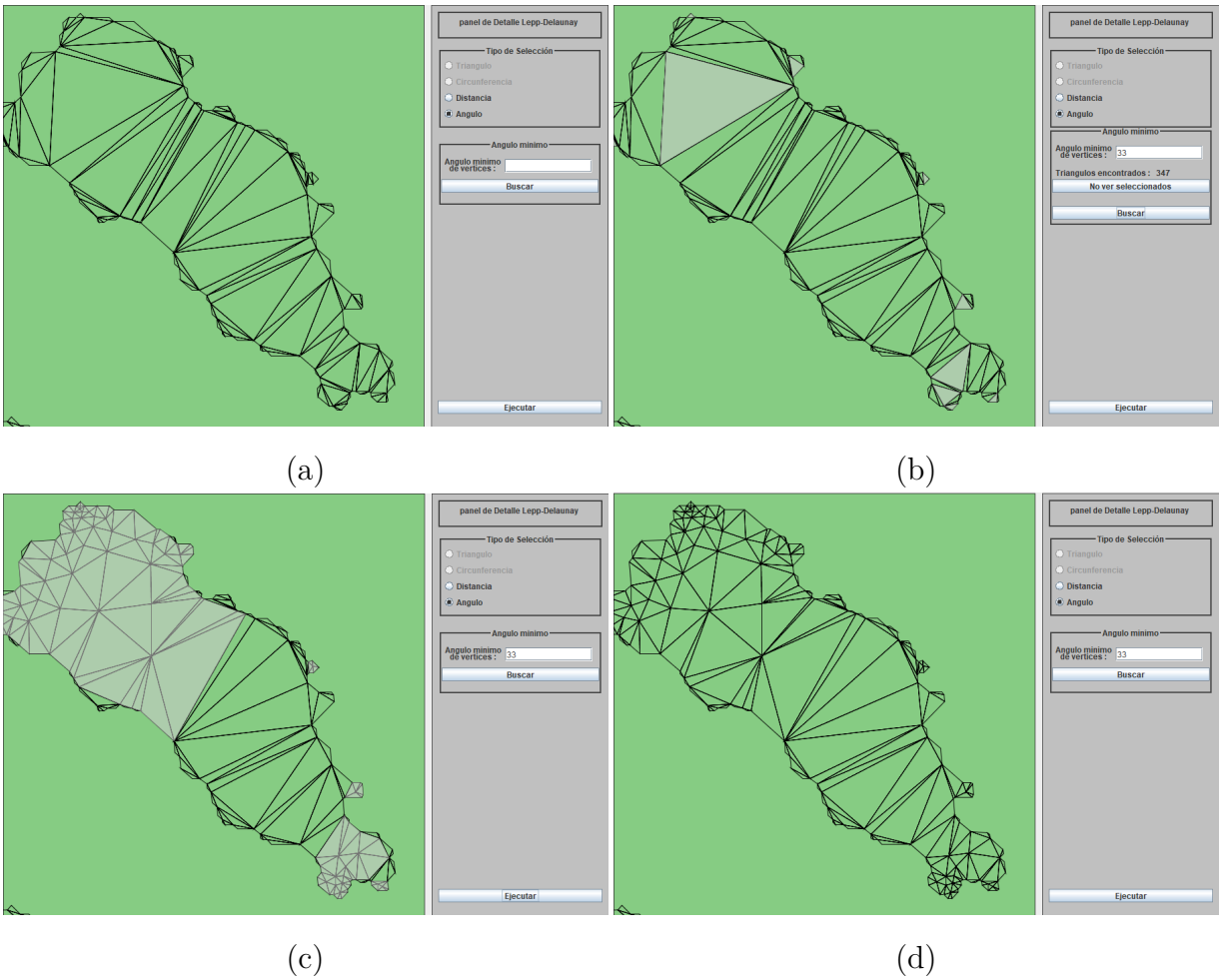


Figura 5.21: Aplicación del algoritmo LEPP-Delaunay para refinar los triángulos seleccionados.

Capítulo 6

Conclusiones

El trabajo desarrollado en este proyecto de título permite establecer las siguientes conclusiones finales:

1. Para el problema de la creación de mallas de triángulos es necesario estructurar la información, y en nuestro caso se creó un modelo, el que estructura una triangulación (Ver Clase 2.1). Un modelo se compone de vértices, aristas y triángulos, estos son necesarios para los algoritmos de refinamiento de mallas.

Una malla de triángulos es un conjunto de n-vértices, los cuales se unen mediante aristas que no se cruzan. Las mallas se pueden clasificar en, mallas estructuradas y mallas no estructuradas.

2. Para solucionar el problema del diseño del software se utilizó el patrón de diseño modelo-vista-controlador, puesto que se cuenta con una parte que el usuario puede ver (vista) un modelo de datos para guardar la triangularización (modelo) y un componente que gestione la funcionalidad del software (controlador), además se utilizó el lenguaje unificado de modelado (UML), específicamente diagrama de clases, para

explicar los componentes del patrón de diseño.

Para explicar la funcionalidad del sistema se utilizó un diagrama de casos de usos y a su vez se crearon tablas de descripción de casos de uso para describir los requerimientos funcionales del sistema.

3. Para el refiminamiento de mallas, se utilizaron los siguientes algoritmos de refinamiento y mejoramiento de mallas basados en los conceptos de LEPP y arista terminal.

Leep-Bisección: Consiste en dividir el triángulo t_n por la arista mas larga en dos nuevos triángulos y así hasta dividir el triangulo t_0.

Leep-Centroide: Consiste en encontrar el centroide del cuadrilátero definido por los triángulos terminales t_{n-1} y t_n, luego de esto por cada uno de los vértices definidos por los triángulos terminales, generar aristas hasta el centroide, lo cual genera cuatro nuevos triángulos.

Leep-Delaunay: Consiste en encontrar el centroide del cuadrilátero definido por los triángulos terminales t_{n-1} y t_n, luego de esto por cada uno de los vértices definidos por los triángulos terminales, generar aristas hasta el centroide, lo cual genera cuatro nuevos triángulos, y por último a estos se les aplica el algoritmo inserción Delaunay, para dejar una malla conforme, hasta dividir el triangulo t_0.

4. Para las pruebas de funcionalidad se utilizaron pruebas locales de los algoritmos y su comportamiento en la gráfica.

Referencias

[1] Boyd, Stephen and Vandenberghe, Lieven. *Convex optimization.* Cambridge university press, 2004.

[2] Platform, Java. *Standard Edition 8 API Specification.* Oracle, 2015.

[3] Baumgart, Bruce G. *Winged edge polyhedron representation.* Tech. rep., Stanford University California Department of Computer Science, 1972.

[4] Constanda, C and Ahues, M and Largillier, A. *Integral Methods in Science and Engineering.* Springer, 2004.

[5] Frey, Pascal Jean and George, Paul L. *Mesh generation: application to finite elements.* Wiley Online Library, 2000.

[6] Leyva, Fernando Cervantes and Becario. *Metodo de diferencis finitas.* 2005.

[7] Garcia, Francisco and Palacio, Carlos and Garcia, Uriel. *Unestructured mesh generation for numeric models implementation.* DYNA 76, 157 (2009), 17–25.

[8] Gutierrez, Claudio and Gutierrez, Flavio and Rivara, Maria-Cecilia. *Complexity of the bisection method.* Theoretical Computer Science 382, Elsevier, 2 (2007), 131–138.

[9] Díaz, A. *Métodos de mallado y algoritmos adaptativos en dos y tres dimensiones para la resolución de problemas electromagnéticos cerrados mediante el método de los elementos finitos.* Valencia: Publicaciones de la Universidad de valencia, 2000.

[10] Rivara, M. C. *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques.* International journal for numerical methods in Engineering 20, 4 (1984), 745–756.

[11] 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 (1997), 3313–3324.

[12] Rivara, M. C. *Lepp-bisection algorithms, applications and mathematical properties.* Applied Numerical Mathematics 59, 9 (2009), 2218–2235.

[13] Rivara, M.-C., and Hitschfeld, N. *Lepp-delaunay algorithm: a robust tool for producing size-optimal quality triangulations.* In IMR (1999), Citeseer, pp. 205–220.

[14] Rivara, M.-C., and Palma, M. *New lepp-algorithms for quality polygon and volume triangulation: Implementation issues and practical behavior.* Asme applied mechanics division-publications-AMD 220 (1997), 1–8.

[15] Rosenberg, I. G., and Stenger, F. *A lower bound on the angles of triangles constructed by bisecting the longest side.* Mathematics of Computation 29, 130 (1975), 390–395.

[16] Swokowski, Earl William and Abreu, José Luis and Olivero, Martha and others. *Calculus with analytic geometry.* 1989.

[17] Xamán, J. *Dinmica de fluidos computacional para ingenieros*. Palibrio, 2016.

[18] Schildt, Herbert. *Java 2: the complete reference*. McGraw-Hill Professional, 2000.

[19] Rivara, Maria-Cecilia and Rodríguez, Pedro and Montenegro, Rafael and Jorquera, Gaston. *Multithread parallelization of lepp-bisection algorithms*. Applied Numerical Mathematics, 62(4):473–488, 2012.

[20] Rodríguez, Pedro and Rivara, María Cecilia and Scherson, Isaac D. *Exploiting the memory hierarchy of multicore systems for parallel triangulation refinement*. Parallel Processing Letters, 22(03), 2012.

[21] Rivara, Maria-Cecilia and Rodríguez, Pedro. *Tuned Terminal Triangles centroid Delaunay algorithm for quality triangulation*. In Proceedings, 2774 International Meshing RoundTable, Albuquerque, New Mexico, USA, 2018.

[22] Rodríguez, Pedro and Rivara, María Cecilia. *Rivaraelaxed Lepp-Delaunay algorithms for the refinement/improvement of triangulations*. In Proceedings of 25 h1, International Meshing RoundTable, Crystal City, Washington DC, USA, 2016.

Apéndice A

Glosario, Siglas y Abreviaciones

A.1. Glosario

Definición A.1 ***Símplex*** *En geometria un símplex o n-símplex (o simplices) es la envoltura convexa de un conjunto de (n+1) puntos independientes en el espacio euclídio de dimencion n o mayor.(Ver figura A.1)*

Un n-símplex estándar es el subconjunto de $\mathbb{R}^{n+1}$ *dados por [1]:*

$$\triangle^n = \left\{(t_0, ..., t_n) \in \mathbb{R}^{n+1} \mid \Sigma_i t_i = 1, t_i \geq 0; \forall i\right\}$$

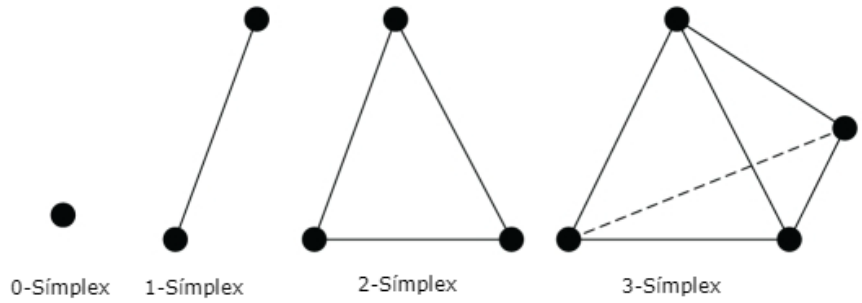


Figura A.1: Representacion de n-símplex [1]

A.2. Siglas y Abreviaciones

- ***CAD:*** Diseño asistido por ordenador.
- ***MEF:*** Método de elementos finitos.
- ***UML:*** Lenguaje unificado de modelado.
- ***MVC:*** Modelo vista controlador.
- ***M2D:*** Archivo de modelado en dos dimensiones.

- ***Neighbors:*** Triángulos vecinos (comparten una arista).

- ***applet:*** Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo, en un navegador web.

- ***JComponent's:*** Es la base de la clases de algunos componentes de java Swing .

- ***LEEP:*** Refinamiento de triangulaciones por la arista mas larga (Longest edge propagation path).

- ***Arista terminal:*** Corresponde al termino de un Leep, en el cual dos triangulos adyacentes comparten su arista mas larga.

- ***Bisección:*** división de un triángulo en dos o mas triángulos según corresponda.

- ***Espacio Euclídeo:*** El espacio euclídeo es un tipo de espacio geométrico donde se satisfacen los axiomas de Euclides de la geometría.