Resumen

Los códigos comerciales de elementos finitos están diseñados con una amplia variedad de herramientas y opciones para satisfacer las necesidades de muchas aplicaciones diferentes. Sin embargo, en un entorno de enseñanza, esto no siempre es deseable, ya que el esfuerzo requerido para aprender el software requiere atención y tiempo de la enseñanza de los conceptos teóricos. El uso de solucionadores completamente desarrollados también podría permitir la resolución de ejercicios sin comprender completamente los conceptos subyacentes a medida que el software los maneja por usted. Para abordar esto, CALFEM (Aprendizaje Asistido por Computadora del Método de Elementos Finitos) fue desarrollado en LTH. CALFEM proporciona una caja de herramientas para resolver problemas de elementos finitos pero con una clara conexión entre los métodos utilizados y los conceptos teóricos rectores. En la actualidad, CALFEM está completamente implementado en MATLAB, pero desde hace unos años también se está desarrollando una versión de Python.

En el método de elementos finitos una parte fundamental del problema es la geometría, la cual se utiliza para generar la malla de elementos finitos. A partir de ahora, para producir una geometría en CALFEM para Python, todos los puntos y las líneas y superficies de conexión deben definirse manualmente en el código. Este es un trabajo tedioso y se limita a una geometría simple con pocos puntos antes de volverse difícil de manejar. Debido a esto, los estudiantes pueden tener dificultades y pasar mucho tiempo trabajando para definir correctamente los puntos geométricos y las conexiones. Para abordar este problema, el propósito de esta tesis ha sido implementar un editor de geometría interactivo diseñado para crear geometrías simples de una manera intuitiva.

El editor ha sido desarrollado en Python combinado con una interfaz gráfica de usuario diseñada en PyQt5. Las funciones implementadas permiten la creación y edición de superficies poligonales y rectangulares en un QGraphicsScene. Los objetos geométricos creados en QGraphicsScene se pueden transformar en un objeto de geometría CALFEM. A partir de esta etapa se puede generar y exportar una malla CALFEM. La funcionalidad principal del editor está desarrollada para su uso en Python, pero también se ha incluido la opción para exportar los resultados de la malla a MATLAB. Para lograr el objetivo de crear un programa simple e intuitivo, la interfaz de usuario se implementó utilizando un diseño similar a una cinta para manejar los controles. La funcionalidad en la ventana del programa se basó en la idea de manipulación directa.

Agradecimientos

El trabajo de esta disertación de maestría se llevó a cabo en la División de Mecánica Estructural de LTH. Quisiera agradecer a mis supervisores Jonas Lindemann y Karin Forsman por su guía y apoyo durante el transcurso del proyecto y por proporcionar valiosos medios de comunicación entre las restricciones durante la pandemia de Covid-19. Agosto 2021, Lund.

1. Introducción

En esta tesis se ha desarrollado un programa para crear y modificar geometrías para ser utilizado en la biblioteca CALFEM del método de elementos finitos (FEM) para Python [1]. Este informe cubre el trabajo del proyecto comenzando con la base de por qué se necesita el programa y para qué propósito debe servir. Se espera que el lector tenga conocimientos básicos sobre el desarrollo de

programas y el método de elementos finitos. En este capítulo se describe la metodología de cómo se realizó el trabajo y qué limitaciones se establecieron para el proyecto.

1.1 Antecedentes

1.1.1 CALFEM

CALFEM (Computer Aided Learning of the Final Element Method) [2] es una biblioteca desarrollada en la Universidad de Lund con el propósito de ayudar a la enseñanza del método de elementos finitos en la Universidad. La idea de la biblioteca es proporcionar una caja de herramientas para resolver problemas de elementos finitos en un entorno pedagógico, brindando una conexión clara entre los conceptos teóricos del modelo de elementos finitos y los métodos utilizados al programar [3]. El desarrollo comenzó ya en los años 70 y ha consistido en múltiples lenguajes de programación. El núcleo de CALFEM hoy en día está diseñado en MATLAB [4], pero desde hace unos años también se ha desarrollado una versión de Python (clonada de la versión existente de MATLAB) [5]. Hay muchos beneficios de usar Python [6], uno de ellos es que Python es de código abierto y no requiere licencias costosas a diferencia de MATLAB. Además, la sintaxis de Python obliga al usuario a escribir un código legible adecuado. Una ventaja de usar MATLAB es el fácil uso de matrices y operaciones con matrices, lo cual es de gran importancia en el método de elementos finitos que se basa en el uso de matrices. Sin embargo, desde el lanzamiento de NumPy para Python [7], herramientas similares están disponibles en Python.

1.1.2 Geometría y creación de mallas

Un paso importante en el método de elementos finitos es la generación de una malla de elementos finitos. Para producir una malla, primero se debe definir la geometría en el espacio y asignarle condiciones de contorno. En CALFEM no existe una interfaz de usuario que permita al usuario crear y modificar geometrías de una manera eficaz e intuitiva, sino que los puntos y líneas de la geometría deben definirse manualmente en el código. Para CALFEM en Python, se ha desarrollado una clase de malla [8] usando Gmsh [9] para realizar la malla, pero la geometría debe definirse manualmente en el código, lo cual es un enfoque engorroso. La producción manual de las geometrías introduce una complejidad innecesaria para ciertos ejercicios en los cursos FEM y puede desviar la atención de otros aspectos clave más importantes de la enseñanza de los conceptos básicos del método de elementos finitos. Esto exige la necesidad de una interfaz gráfica de usuario que permita a los estudiantes, de manera sencilla, crear las geometrías necesarias en los ejercicios. El uso de una interfaz gráfica de usuario permite un proceso de trabajo diferente en comparación con el uso exclusivo de secuencias de comandos, en particular, proporciona una entrada directa al dibujar geometrías.

Para satisfacer la necesidad de una interfaz gráfica de usuario para crear geometrías, se podría usar un software de terceros, por ejemplo, el generador de mallas y geometrías de la biblioteca pdetool [10] se puede usar en MATLAB. Sin embargo, el inconveniente de este software es que a menudo se crean como programas profesionales dirigidos a ingenieros en análisis de elementos finitos con una gran cantidad de funciones y herramientas y con un enfoque en la eficiencia y la flexibilidad. Esto generalmente significa que los programas tienen una curva de aprendizaje alta y el usuario

debe pasar mucho tiempo aprendiendo el programa antes de poder usarlo correctamente. Para los estudiantes, usar este tipo de software requeriría mucho tiempo y muy poco beneficio.

1.2 Objetivo

El objetivo del proyecto es desarrollar una interfaz gráfica de usuario (GUI) con herramientas para dibujar geometrías 2D, integrada con Gmsh y CALFEM para Python. Como requisitos adicionales, la GUI debe ser fácil e intuitiva de usar sin necesidad de una capacitación extensa y debe estar dirigida a un entorno educativo de nivel universitario. El programa también debe ser versátil para ser utilizado en tres escenarios diferentes:

1. Como función en CALFEM para Python donde se abre el editor y permite crear o editar geometrías antes de cerrar y continuar con el script. Para permitir la interactividad, las geometrías preexistentes deberían poder pasarse a la función con la opción de edición.

2. Como un programa independiente con posibilidades de dibujar geometrías, agregue condiciones de contorno y genere resultados de malla que se pueden guardar para su uso posterior tanto en CALFEM para MATLAB como en CALFEM para Python.

3. Como componente integrable en otras interfaces gráficas basadas en Qt para ser utilizado como recurso en cursos de la universidad o en futuros programas.

1.3 Método

La idea inicial del programa era usar la biblioteca QGraphicsScene [11] de PyQt5 [12] y utilizar las diferentes formas disponibles: líneas, elipses, rectángulos y polígonos. Las implementaciones de estas clases de elementos se usarían para dibujar y crear la geometría, así como para rastrear los componentes en QGraphicsScene.

Se exploraron y compararon varios paquetes de software diferentes para evaluar qué características y representaciones se consideraron relevantes para usar como inspiración.

Se estableció una línea base de la funcionalidad principal del programa. Además, se agregaron limitaciones para limitar el alcance del trabajo y garantizar la simplicidad del programa. Luego, el trabajo comenzó con la implementación de la funcionalidad requerida. A medida que avanzaba el trabajo, se llevaron a cabo reuniones recurrentes para discutir el estado del proyecto y cómo proceder, creando un proceso iterativo. Una vez terminado el núcleo del programa, se reestructuró y diseñó la interfaz gráfica de usuario teniendo en cuenta la facilidad de uso y la simplicidad.

1.4 Limitaciones

Durante el curso del proyecto hubo que hacer algunas limitaciones para reducir el alcance del trabajo con el fin de ajustarlo dentro del marco de tiempo disponible. También se incluyeron algunas limitaciones para garantizar que el programa siguiera siendo simple y fácil de usar.

1.4.1 Geometrías bidimensionales

Como los cursos introductorios de FEM en la universidad se ocupan principalmente de problemas bidimensionales, el programa se limitó a dos dimensiones. Agregar profundidad para crear geometrías tridimensionales aumenta enormemente la complejidad del programa de dibujo y complica la visión general del problema para el usuario. Por lo tanto, limitarlo a dos dimensiones reduce la complejidad del uso del programa.

1.4.2 Condiciones de contorno

En el método de elementos finitos, un componente clave para producir modelos precisos son las condiciones de contorno. Al crear geometrías en mecánica, ejemplos de estas condiciones son el soporte de rodillos y el soporte fijo. Sin embargo, el programa está destinado a ser de uso general y ser capaz de manejar otras áreas problemáticas, como la transferencia de calor y el flujo de agua subterránea. Cada una de estas áreas tiene diferencias tales como números de grados de libertad y unidades de medida. Por lo tanto, se determinó que implementar una interfaz que cubriera todas estas posibilidades sería demasiado complejo. En cambio, la aplicación amplía la funcionalidad de CALFEM para Python, al permitir que el usuario establezca marcadores en los bordes y vértices de destino. Luego, los marcadores se guardan junto con la geometría para permitir que el usuario maneje las condiciones de contorno más adelante en el código. Esto también está en línea con los principios de CALFEM, lo que permite al usuario tener una mejor interpretación de la implementación [3]. Además, elimina la necesidad de usar el programa para ajustar los valores de las condiciones, lo que puede facilitar el ajuste y probar diferentes valores.

1.4.3 Formas Geométricas

La idea inicial era utilizar las cuatro formas geométricas: líneas, elipses, rectángulos y polígonos. Sin embargo, debido a decisiones tomadas durante el progreso, la implementación se limitó a usar solo polígonos y rectángulos como un caso especial de polígono. Para una discusión más detallada, consulte la Sección 5.1.2.

2 Revisión de herramientas existentes

Para implementar un programa con el propósito de crear geometrías e integrar los componentes necesarios para ser utilizado en un entorno FEM, se inspiró en otros programas similares. Esto para adquirir conocimientos e ideas sobre qué características y herramientas de las aplicaciones establecidas pueden ser buenas para usar y son fáciles de entender de una manera intuitiva, pero también para encontrar qué se debe evitar y qué se puede mejorar o simplificar.

2.1 PowerPoint

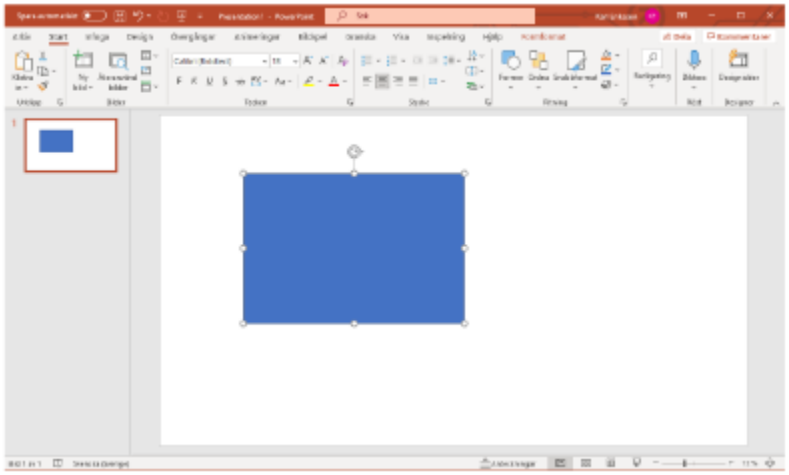

Figura 2.1: Ejemplo de instantánea del trabajo con geometrías en PowerPoint

Se utilizó PowerPoint [13] como fuente de inspiración de cómo dibujar superficies de una manera fácil de usar e intuitiva. PowerPoint es una herramienta para hacer presentaciones, que se muestra en la Figura 2.1, con geometrías de dibujo solo como una pequeña subherramienta. Por lo tanto, sirve como un buen punto de vista de cómo hacer una herramienta de dibujo simple e intuitiva, ya que no se espera que el usuario tenga experiencia previa. Se exploró cómo se crean las formas básicas, como rectángulos y círculos, en el programa. Además, se examinó cómo se manejan estas formas al moverlas, escalarlas y remodelarlas. Los hallazgos se usaron como ideas de cómo implementar una funcionalidad similar.

2.2 Pdetool

Pdetool es parte de la caja de herramientas de ecuaciones diferenciales parciales (PDE) en MATLAB [10] y funciona como un programa independiente que se inicia desde la línea de comandos de MATLAB. El programa sirve como un solucionador completo de numerosos problemas de PDE, pero en este trabajo solo se consideraron las herramientas de creación de geometrías y mallas al explorar sus funcionalidades. Pdetool se utiliza como complemento de la biblioteca CALFEM para MATLAB para crear geometrías y mallas en algunos cursos FEM en la Universidad de Lund. El programa está definido para un dominio 2D y el creador de geometría, que se muestra en la Figura 2.2, se basa en la combinación de objetos gráficos simples. Las formas disponibles son: rectángulo, círculo, elipse y polígono. Estas formas se mueven y giran para formar grupos superpuestos. A cada forma se le asigna una variable y una fórmula global rige cómo deben combinarse las formas para formar la geometría; Un signo positivo agrega la forma a la geometría combinada mientras que un signo negativo resta la forma produciendo un agujero o un corte dependiendo de la superposición.

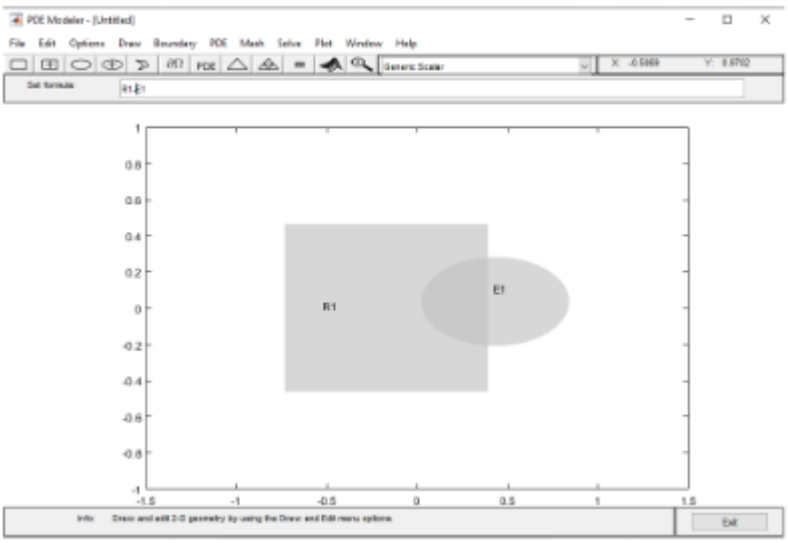

Figura 2.2: Ejemplo de instantánea del trabajo con geometrías en Pdetool

Ventajas:

• Con la fórmula es muy fácil crear agujeros y cortar partes de un objeto.

• Las herramientas muy básicas se presentan de forma sencilla y las opciones más avanzadas, p. la malla está disponible a través del menú, pero no es algo por lo que el usuario sin experiencia deba preocuparse.

• El uso de hacer clic con el botón derecho para terminar un polígono es útil, ya que no tiene que depender de la precisión del usuario para hacer clic cerca del punto de inicio para cerrar la superficie.

Contras:

• La agrupación y selección de elementos puede resultar confusa, p. al hacer clic en geometrías superpuestas, se selecciona todo lo que está detrás del cursor. Al hacer clic de nuevo, la nueva selección depende de si la nueva selección incluye alguno de los elementos ya seleccionados o no. Para restablecer completamente la selección, primero se debe hacer clic en el fondo.

• Al alternar entre las diferentes vistas, no hay una indicación clara de en qué vista se encuentra actualmente (excepto los efectos gráficos en la geometría, pero no el título o similar).

• La fórmula a veces se puede experimentar como inconsistente, es decir, (R1 + R2) - R3 no es lo mismo que R1 + R2 - R3

• Usar solo geometrías superpuestas es intuitivo pero puede requerir mucho esfuerzo para encontrar una buena manera de construir una geometría más compleja. Para geometrías complejas, puede requerir una gran cantidad de elementos que se superponen. Esto hace que sea más difícil obtener una visión general adecuada con todas las geometrías superpuestas. También puede hacer

que sea más difícil seleccionar el elemento o elementos correctos con la forma en que funciona la selección.

• La información sobre herramientas es buena, pero está ubicada en la esquina inferior izquierda, lo que la hace menos perceptible, ya que el foco suele estar alrededor del lienzo de dibujo y la barra de herramientas.

2.3 GUI de Gmsh

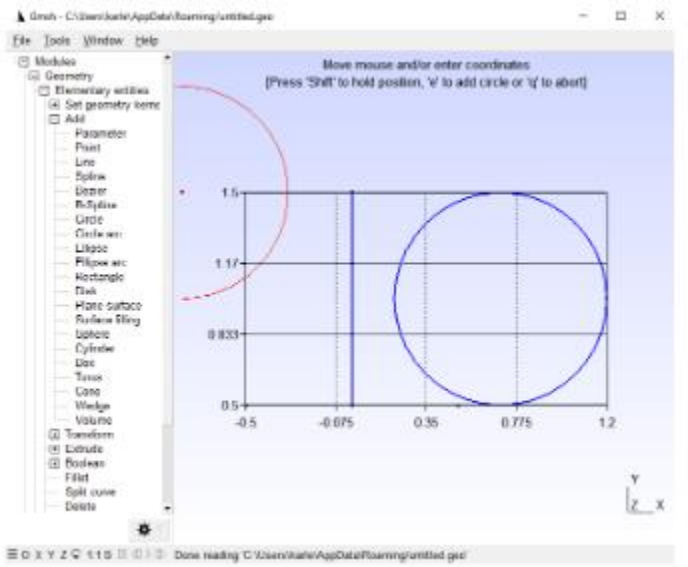

Figura 2.3: Ejemplo de instantánea del trabajo con geometrías en la GUI de Gmsh Gmsh [9], que es la biblioteca de mallas utilizada en CALFEM.

Tiene su propia GUI, que se muestra en la Figura 2.3, pero tras la inspección hay algunos inconvenientes. La GUI no es muy intuitiva para usuarios sin experiencia y resulta difícil de manejar sin tener que aprender primero la GUI en detalle. En segundo lugar, está diseñado para manejar geometrías 3D, lo que a su vez complica el uso de geometrías 2D simples. Además, se basa en el uso de la biblioteca Gmsh completa, es decir, se incluye una gran cantidad de opciones y funciones diferentes, lo que complica aún más la comprensión para un usuario sin experiencia.

3 Descripción general del editor de geometría CALFEM

Este capítulo ofrece una breve descripción general de la estructura del Editor de geometría CALFEM al describir las clases subyacentes y cómo interactúan. Se describe el modelado de geometría para explicar el manejo de los elementos geométricos dibujados en PyQt5 y la conexión con la geometría generada usando CALFEM. Además, el diseño se resume describiendo los controles de la interfaz de usuario y cómo gobiernan las funciones utilizadas. Para obtener descripciones más detalladas de la interfaz de usuario y los comandos, consulte el Apéndice A.

3.1 Interfaz de usuario

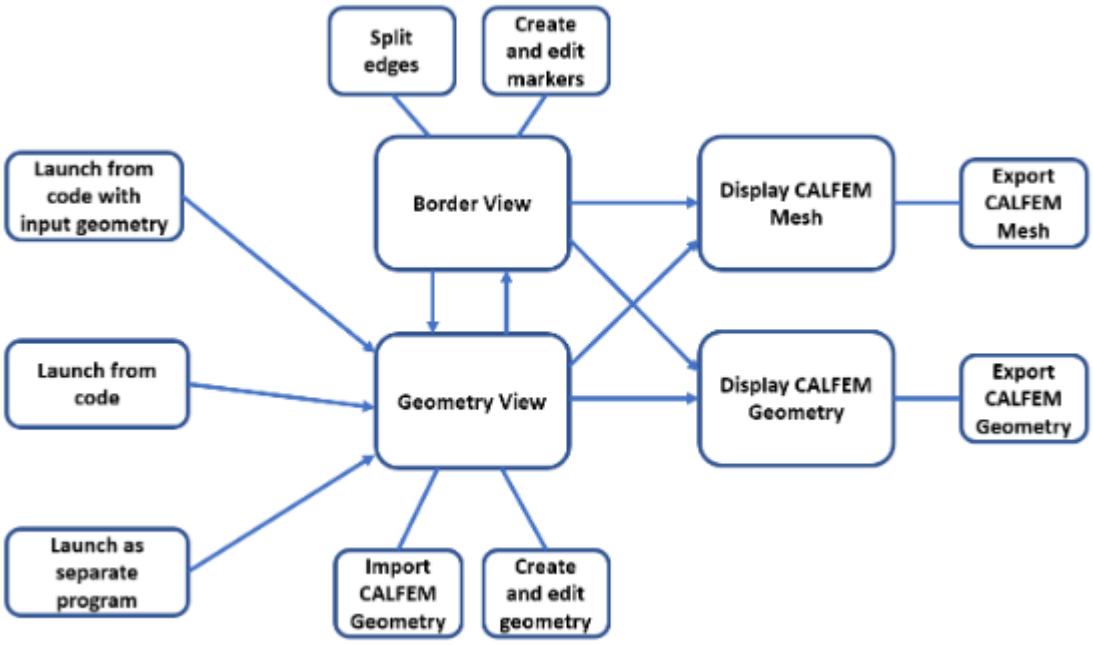

Figura 3.1: Diagrama de las conexiones entre las diferentes vistas y herramientas en la interfaz de usuario

En la Figura 3.1 se muestra el flujo de trabajo de las vistas y herramientas del programa. El programa se puede iniciar con tres opciones diferentes: desde comando, desde comando con una geometría preexistente como argumento y como programa independiente. El programa siempre se inicia en la vista de edición de geometría, la Figura 3.2 muestra un ejemplo del programa iniciado sin ninguna geometría preexistente. Desde allí es posible, seleccionando herramientas en la cinta, cargar geometrías desde un archivo, agregar geometrías dibujando o editar objetos de geometría existentes. Presionar una pestaña cambia el modo del programa y cambia automáticamente las herramientas disponibles para el modo seleccionado. Las otras opciones de edición disponibles se encuentran en la vista de borde, en la que solo se muestran los bordes de los objetos gráficos. Allí, se pueden agregar marcadores a líneas o puntos y las líneas se pueden dividir para agregar nodos adicionales a lo largo de una línea existente. Cuando finaliza la creación y edición de una geometría, se puede generar y mostrar como un objeto de geometría CALFEM o como un objeto de malla CALFEM. Si la malla o la geometría son satisfactorias, se pueden exportar o se devolverán en código según el comando utilizado para iniciar el programa.

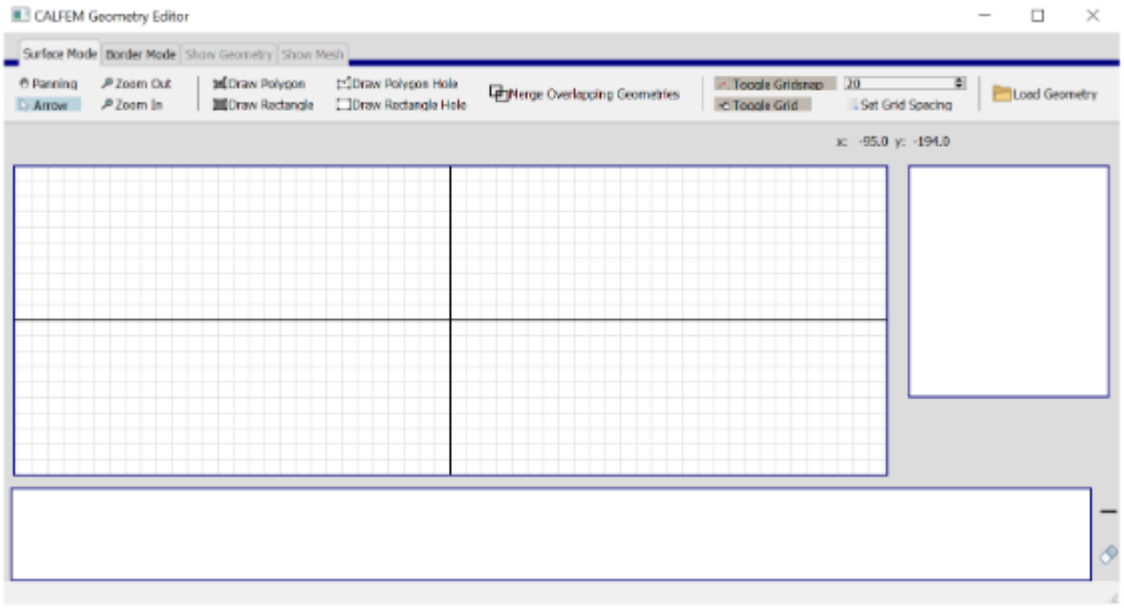

Figura 3.2: Instantánea del programa Editor de Geometría CALFEM en el momento del lanzamiento, antes de que se haya realizado ninguna acción

3.2 Descripción de la clase

Esencialmente, el programa está implementado por tres clases diferentes: EditorWindow, EditorScene y CALFEM para Python, que se ven en la Figura 3.3. EditorWindow extiende la estructura nativamente llamada MainWindow en PyQt5, de manera similar EditorScene extiende QGraphicsScene desde PyQt5. La importación y la exportación son conjuntos de métodos recopilados para manejar el guardado y la carga de datos en el programa o scripts. La exportación se implementa utilizando la biblioteca pickle de Python [14] para almacenar los datos deseados en un archivo apropiado, con la excepción de guardar en MATLAB, que utiliza la función "savemat" de SciPy [15]. Luego, la importación se estructura para desempaquetar el archivo pickle en el orden variable correspondiente y escribir como la exportación. Tanto MainWindow como GraphicsScene se implementan utilizando PyQt5 junto con una gran cantidad de métodos para manejar la creación y edición de las geometrías. Están estrechamente entrelazados ya que MainWindow maneja los controles en los que se realizan los métodos en GraphicsScene. A su vez, MainWindow se actualiza según las acciones realizadas en GraphicsScene. La geometría creada por el usuario en GraphicsScene luego se procesa para recrear la geometría como un objeto de geometría CALFEM. Este objeto de geometría puede usarse como entrada a la clase de malla CALFEM para generar un objeto de malla y las matrices de datos correspondientes.

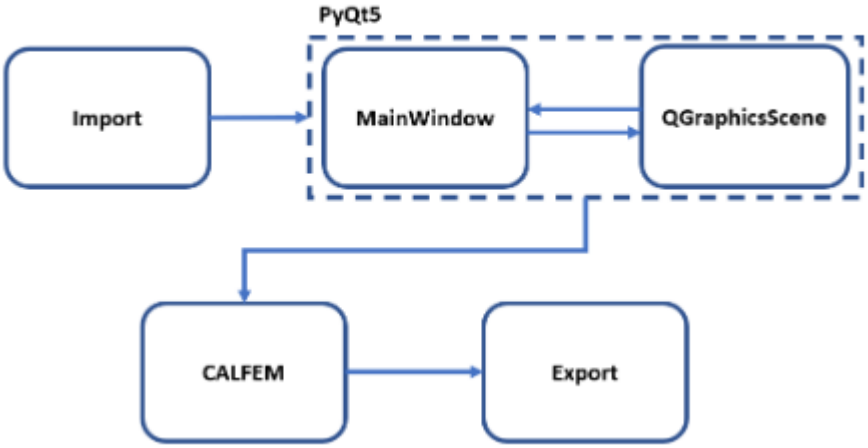

Figura 3.3: Diagrama de las clases utilizadas en el programa

3.3 Modelado Geométrico

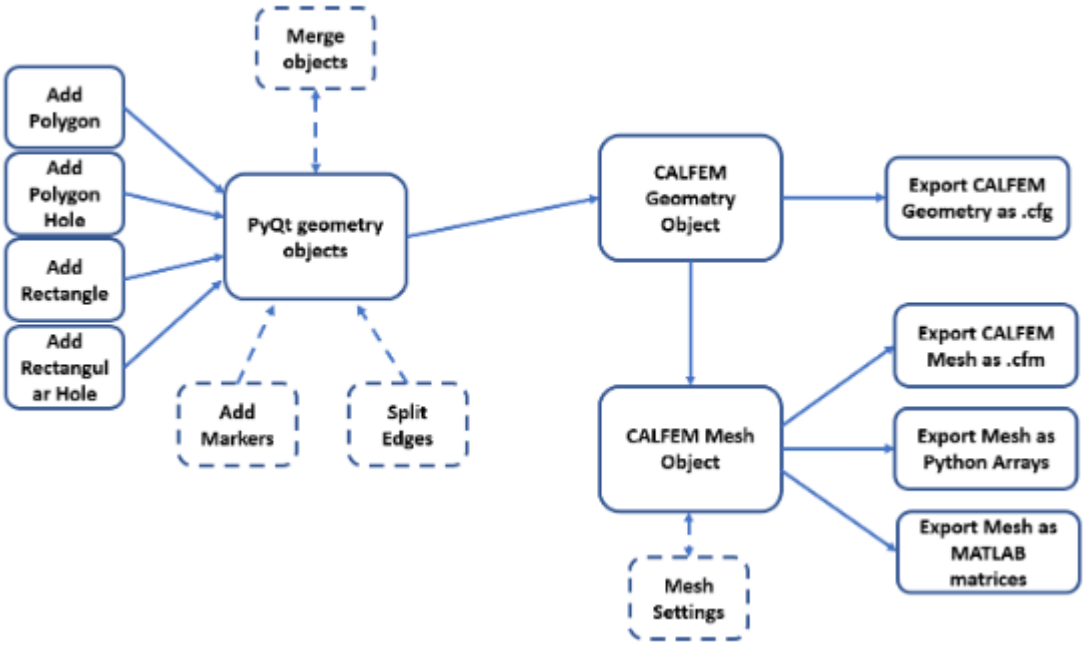

Figura 3.4: Cómo se realiza el manejo de los objetos de geometría en el programa, las flechas discontinuas muestran acciones que no agregan nuevos objetos sino que alteran los ya existentes

Los objetos geométricos creados por el usuario en GraphicsScene se hacen usando funciones de PyQt5 y, en cierta medida, manejados por las funciones nativas correspondientes. Sin embargo, algunos atributos tuvieron que ser creados manualmente para rastrear ciertas variables necesarias para la aplicación. En consecuencia, algunos métodos en GraphicsScene, como hacer clic y mover el mouse, se anulan en la clase para manejar ciertos eventos.

Sin embargo, los objetos creados y editados por el usuario en la escena son objetos derivados de PyQt5.

Para crear una malla a partir de la geometría de la escena, debe transformarse de objetos PyQt5 a la estructura utilizada para describir geometrías en CALFEM para Python. Esto se hace creando una instancia de geometría CALFEM y luego iterando a través de todos los objetos creados en GraphicsScene. Todos los objetos se convierten a la estructura de objetos de CALFEM Geometry adecuada. Entonces es posible guardar la geometría generada o utilizar las funciones CALFEM para generar una malla a partir de esta geometría.

4 Implementación del editor de geometría CALFEM

La implementación del Editor de Geometría CALFEM se detalla en este capítulo. Se describen las bibliotecas subyacentes utilizadas en la implementación. Dado que describir todas las funciones del programa sería demasiado extenso, se han seleccionado algunas funciones básicas. Se considera que estas funciones básicas representan los aspectos más importantes de la implementación y requieren una explicación para proporcionar una comprensión suficiente. Además, se explican los pensamientos detrás del diseño de la GUI.

4.1 Bibliotecas subyacentes

4.1.1 CALFEM y Gmsh

En general, se realizan una serie de pasos para realizar cálculos FEM simples. Estos pasos consisten en [5]:

1. Definir el modelo

2. Generar matrices de elementos

3. Ensamblar matrices de elementos en el sistema global de ecuaciones

4. Resuelve el sistema global de ecuaciones

5. Evaluar las fuerzas de los elementos

Los pasos 1. , 2. y 3. son los pasos considerados cuando se trata de la generación de geometría y malla. La geometría debe estar definida para generar una malla. La malla entonces contiene todos los elementos y las matrices correspondientes. Estas matrices también proporcionan la estructura en la que los elementos se conectan entre sí para formar la geometría y las condiciones de contorno. En CALFEM, el mallado se realiza con la biblioteca Gmsh. Gmsh [9] es un generador de malla de elementos finitos de código abierto rápido y robusto, disponible en varios lenguajes de programación, incluido Python. Por lo tanto, la entrada de geometría a la clase de malla CALFEM se basa en el manejo de la geometría de Gmsh. Al producir una malla, se devuelve una serie de matrices y diccionarios:

• Dof: matriz de todos los grados de libertad totales que se encuentran en la geometría, cada fila de la matriz representa un nodo y sus grados de libertad correspondientes forman las columnas. Por lo tanto, la dimensión de la matriz es [número de nodos x grados de libertad por nodo].

• Coords - Array que contiene las coordenadas para cada grado de libertad, consta de dos o tres columnas dependiendo si la geometría es 2D o 3D. En este trabajo solo se considera 2D y la dimensión del arreglo [número de elementos x 2].

• Edof: los grados de libertad de los elementos conectan cada elemento con sus nodos y sus grados de libertad. El número de columnas depende tanto del tipo de elemento como del número de grados de libertad por nodo, es decir, el caso estándar con elemento triangular con dos grados de libertad por nodo da un ancho de 6 (tres nodos por elemento con dos grados de libertad cada uno). La fórmula general para la dimensión del arreglo es [número de elementos x (nodos por elemento x grados de libertad por nodo)].

• Bdof - Diccionario de todos los grados de libertad a lo largo de los límites. De forma nativa, todos los grados de libertad se asignan en una lista a la entrada 0 del diccionario si se proporcionan bordes marcados. Si se asigna un marcador a un borde o punto de borde, el nombre del marcador de ese punto o borde se agrega como una clave en el diccionario y una lista de los grados de libertad correspondientes como valores.

• Marcadores de elemento: lista donde cada entrada corresponde al marcador de elemento del elemento de ese número de índice

• Elementos de límite: argumento opcional que proporciona un diccionario de todos los elementos a lo largo de los bordes

4.1.2 PyQt y Qt

Qt es una biblioteca de diseño gráfico de usuario basada en C++ [16]. Una de las principales ventajas de Qt es la compatibilidad multiplataforma. Independientemente del sistema operativo de desarrollo, la aplicación producida se puede usar en muchos escritorios diferentes, plataformas integradas y móviles. Lo más notable es el uso en Windows, OS X y Linux. Qt se basa en el uso de diferentes tipos de widgets, como botones y cuadros de texto. Los widgets se agregan a la aplicación y se ordenan mediante señales, los diseños se utilizan para controlar la posición y distribución de los widgets dentro de la aplicación. Los terceros también permiten extensiones gratuitas que permiten el desarrollo en otros lenguajes además de C++, que se utiliza en este proyecto como la extensión PyQt5 [12] de Riverbank Computing que permite el desarrollo en Python.

QGraphicsScene.

La interfaz de usuario de la aplicación está compuesta por varios widgets diferentes de Qt, como QTabWidget y QToolButton. Todos los widgets están conectados entre sí en la clase MainWindow. Sin embargo, el componente más importante es QGraphicsScene, ya que es el componente en el que se realiza la creación y edición de las geometrías. Usando controles separados en QGraphicsScene y anulando las señales nativas como MousePressEvent y MouseMoveEvent, las acciones se controlan en la escena para producir el comportamiento buscado mientras se siguen utilizando muchos de los comandos y objetos nativos de QGraphicsScene.

4.2 Funciones principales

4.2.1 Conversión de objetos gráficos en una geometría Gmsh Descripción

El programa opera realizando el dibujo y movimiento de geometrías completamente separado del código Gmsh. Cuando se completa la edición, el código Gmsh se genera en función de la geometría existente. Cada elemento geométrico agregado se almacena en una lista. Cuando se genera el código Gmsh, esta lista se repite y para cada elemento se agregan los puntos y los bordes de conexión. Para generar una malla correcta se cubren algunos casos especiales para asegurar la correcta interacción entre los elementos geométricos: puntos superpuestos, arista superpuesta de puntos y aristas superpuestas.

Puntos superpuestos

Los puntos superpuestos corresponden a dos elementos geométricos que comparten un nodo común. Por lo tanto, es importante que la geometría Gmsh se genere con este nodo compartido entre los dos elementos. De lo contrario, dos nodos con las mismas coordenadas serían independientes entre sí, por lo que los elementos no funcionarían como una unidad continua. Para manejar esto, se hace una simple verificación de si ya se ha agregado un punto en el código Gmsh en las mismas coordenadas, si es así, no se agrega ningún punto nuevo, sino que se usa el ya existente.

Punto superpuesto en el borde

Por el mismo razonamiento que para los puntos superpuestos, es importante que un punto de esquina de un elemento que se superponga a un borde de otro elemento se conecte a ese borde para garantizar que los cambios en el nodo afecten a ambos elementos. Para manejar esto, se pasa una verificación de todos los elementos antes de la generación de la geometría Gmsh para determinar si hay algún punto superpuesto en alguno de los bordes. Si ese es el caso, se agrega un punto adicional al elemento en el que se superpone el borde. El punto agregado esencialmente divide el borde en dos bordes separados. Dado que ahora hay dos puntos exactamente en la misma posición, se maneja mediante los puntos superpuestos explicados anteriormente.

Bordes superpuestos

Los bordes superpuestos consisten en dos escenarios: todo el borde se superpone (los elementos comparten un borde) o parte del borde se superpone. Si todo el borde se superpone, los puntos de borde de ambos bordes también se superponen. Durante la generación de geometría, uno de los bordes se agregará primero, sin el conocimiento de una superposición. Cuando se va a añadir el segundo borde, se detecta la superposición. En lugar de agregar el segundo borde, se usa el ya agregado. Este borde se usa para conectarse con el resto de la geometría y, por lo tanto, tratar los bordes superpuestos como el mismo borde.

En el caso de aristas parcialmente superpuestas, la solución es la misma que para un punto superpuesto en una arista. Como uno de los bordes es el más corto, los extremos de este borde se superponen al borde del otro elemento. En la posición donde ocurre la superposición, se agregan dos nuevos puntos en el borde. Después de este escenario, se usa el mismo procedimiento como si todo el borde se superpusiera.

4.2.2 Ajuste de puntos a puntos y ajuste de puntos a bordes

La superposición descrita en la sección anterior se basa en que los dos puntos de intersección comparten las coordenadas exactas. A menudo es difícil para los usuarios colocar puntos con píxeles perfectos uno encima del otro. Por lo tanto, se requiere el ajuste de puntos a otros puntos o bordes. El ajuste se habilita cuando se mueve un punto de esquina en la escena.

Ajuste de punto a punto

Cuando el punto seleccionado se mueve en la escena, se compara con una lista almacenada de todos los puntos de esquina de la escena. Usando el comando de norma NumPy, se calcula la distancia euclidiana a todos los demás puntos. Si alguno de los otros puntos (no se compara consigo mismo) está dentro de una distancia de 10 unidades, las coordenadas del punto seleccionado se reemplazan por las coordenadas del punto cercano.

Ajuste de punto a borde

El ajuste de un punto a un borde funciona ligeramente diferente. Aquí no es factible almacenar cada punto de un borde en una lista para comparar. En cambio, cada borde se almacena y cuando se mueve un punto seleccionado en la escena, se usa un área cuadrada de 10x10 píxeles alrededor del puntero. Cada punto del área se comprueba si está contenido en alguno de los bordes. En ese caso, el punto se agrega a una lista temporal. Si hay varios puntos, que suele ser el caso, ya que es probable que el área cuadrada se superponga a más de un píxel del borde, la lista temporal se recorre y el punto con la distancia euclidiana más pequeña al puntero se selecciona como el punto al que ajustar.

Vale la pena señalar que también existe una alternativa para usar el ajuste de cuadrícula. A continuación, el punto se ajusta al punto más cercano de la cuadrícula. Sin embargo, es posible crear primero geometrías con el ajuste de cuadrícula desactivado. Si luego se activa el ajuste a la cuadrícula, es posible que los puntos creados anteriormente no estén en la cuadrícula. Si la alineación de la cuadrícula tuviera prioridad, estos puntos serían inalcanzables mientras la alineación de la cuadrícula esté activada. Por lo tanto, para garantizar las superposiciones correctas, se prioriza de modo que la alineación con otro punto o borde tenga prioridad sobre la alineación con la cuadrícula.

4.2.3 Vistas y modos de modelado

El programa consta de cuatro vistas diferentes que están controladas por un widget de pestaña para garantizar que solo una de ellas esté activada a la vez. El panel de la barra de pestañas también garantiza que solo las herramientas relevantes estén disponibles para la vista activada.

Vista de superficie

La vista de superficie es la vista principal del programa. Esta vista se utiliza para crear, cargar, editar y mover geometrías. También permite dividir los bordes agregando un nodo adicional.

Vista de borde

La vista de borde se utiliza para mostrar solo los bordes de las geometrías y permite editar los marcadores de los puntos y bordes.

Vista de geometría

La vista de geometría utiliza el método CALFEM vis para mostrar la geometría. Esto permite al usuario ver la geometría CALFEM en el programa para asegurarse de que todo funcione según lo previsto.

Vista de malla

De manera similar a la vista de geometría, la vista de malla muestra la malla usando CALFEM vis. La malla se genera utilizando valores estándar, pero los controles permiten al usuario volver a generar la malla utilizando otras configuraciones, como el tamaño máximo del elemento y los grados de libertad por nodo.

4.2.4 Exportar/Importar archivos

En Python La exportación e importación de archivos se implementan en un módulo Python separado como una serie de métodos para permitir su uso independientemente del programa. Para realizar esto, se utiliza la biblioteca Pickle de Python para guardar datos en archivos locales. Con esta herramienta, el comando de exportación escribe los atributos de destino en el archivo en un orden específico y el comando de importación los carga en el orden correspondiente. Los comandos de exportación e importación se realizaron en tres versiones diferentes. Cada versión para manejar una etapa en la que el usuario podría desear guardar su trabajo. Para evitar problemas con los usuarios que cargan entradas no válidas, a las tres versiones se les asignó una extensión de archivo:

• Guardar instancia de geometría CALFEM - .cfg

• Guardar instancia de malla CALFEM - .cfm

• Guardar arreglos de malla - .cfma

En el programa, solo se permite importar objetos de geometría CALFEM (.cfg), ya que la conversión debe volver a realizarse en objetos QtGraphics. Esto fue relativamente simple para los objetos geométricos cuando solo se consideraban polígonos, ya que la estructura es similar con los nodos y las líneas de conexión. Los archivos guardados solo como un objeto de malla CALFEM (.cfm) o como arreglos de malla (.cfma) tienen una estructura diferente que es demasiado compleja para volver a convertirlos en una descripción de la geometría.

En MATLAB

Para exportar los resultados a MATLAB, se adoptó un enfoque ligeramente diferente. Aquí solo los arreglos de la generación de malla son relevantes ya que la versión MATLAB de CALFEM no se implementa en base a Gmsh. Para exportar los arreglos a cargar en MATLAB se utilizó la librería Savemat de SciPy [15]. Esta herramienta convierte las matrices NumPy en matrices MATLAB y el diccionario Python en estructuras MATLAB y las guarda en el formato .mat nativo de MATLAB. Esto produce un archivo que se puede cargar en MATLAB. Cabe señalar que MATLAB no acepta números en nombres de variables, lo que significa que los marcadores de límite proporcionados deben proporcionarse como texto puro para que aparezcan en la exportación.

4.3 Interfaz gráfica de usuario

Para implementar un programa que sea fácil de usar e intuitivo para los nuevos usuarios, la implementación se basó en la idea de manipulación directa [17]. La manipulación directa es una idea de proporcionar al usuario impresiones visuales constantes en respuesta directa a la realización de acciones en el programa. Al dibujar geometrías, esto se hace mostrando nodos y líneas de apoyo durante la creación que se actualizan con el movimiento del mouse. Una vez que se completa la superficie, estos elementos de apoyo se eliminan y se dibuja la superficie real. El proceso de creación de las geometrías sigue un proceso relativamente lineal, donde el usuario crea una geometría, modifica los bordes, genera una geometría CALFEM y si la geometría es satisfactoria genera una malla CALFEM. Sin embargo, no es un orden fijo y se debe permitir al usuario saltar de un lado a otro entre los diferentes modos. Además, solo las herramientas que son relevantes para la vista seleccionada deben estar disponibles para el usuario. Para combinar estos dos requisitos en el programa, se usó una barra de pestañas para manejar los diferentes modos donde hacer clic en una pestaña cambia automáticamente de modo. Cada pestaña también tiene su panel respectivo, en este panel se muestran las diferentes herramientas disponibles y, por lo tanto, solo se muestran las herramientas relevantes para la vista actual.

En el panel, se determinó usar un diseño similar a una cinta, basado libremente en las pautas de Windows [18]. Una cinta es una barra de comandos que reemplaza las barras de herramientas y menús tradicionales. Se enfoca en recopilar herramientas similares en grupos relevantes y está destinado a ayudar a los usuarios a encontrar características y funciones más fácilmente. Una razón para usar una cinta en lugar de una barra de herramientas clásica es que la cinta permite más espacio y texto, así como otros componentes, como un cuadro de número. Al usar una barra de herramientas, algunos de los íconos como "dibujar rectángulo" y "dibujar agujero de rectángulo" pueden verse muy similares y complementar con etiquetas de texto brinda mayor claridad para evitar confusiones al usar el programa. La desventaja es que la cinta ocupa más espacio que una barra de herramientas compacta, sin embargo, dado que el programa siempre necesitará un tamaño de ventana relativamente grande debido a la escena gráfica, el tamaño de la cinta no se considera un obstáculo ya que es relativamente pequeño en comparación.

4.4 Interacción del código del editor

La implementación viene con tres funciones diferentes disponibles que se utilizan para iniciar el editor.

1. ejecutar editor (): inicia el editor como un programa independiente sin ninguna entrada o salida

2. ejecutar editor y cargar (g): inicia el editor con un objeto de geometría CALFEM (g) como entrada, no devuelve salida

3. g, marcador dict = cfe.editar geometría(g): Inicia el editor con la opción de proporcionar un objeto de geometría CALFEM (g) como entrada. Al salir, el editor devuelve un objeto de geometría CALFEM (g) de la geometría, en el momento de la salida, en la escena gráfica del editor. También devuelve un diccionario de marcadores (dictador de marcadores) que contiene los nombres de los marcadores conectados a los índices de los marcadores en el objeto de geometría.

Usando la opción 1 o 2, el usuario tiene que guardar manualmente el trabajo usando las funciones disponibles en la interfaz de usuario. Estas opciones están pensadas para iniciar el editor fuera de un script. Opción 1 al crear geometrías para usar en trabajos futuros que luego se guardan como archivos locales. La opción 2 se usa de manera similar cuando hay un objeto de geometría CALFEM preexistente que el usuario desea editar, la geometría modificada se guarda como un archivo local. La opción 3 es el comando que se usa cuando se llama al editor desde un script, ya que devolverá la descripción de la geometría en el editor directamente al script. Por lo tanto, este comando se usa cuando se realizan cambios menores que se devuelven directamente al script. Aunque, dado que la única diferencia entre las opciones está en cómo se inicia el programa, todas las herramientas y funciones están disponibles en todos los casos. De esta forma, usar la opción 3 permite guardar en archivos locales y volver al script.

5 Discusión y Conclusiones

En este capítulo se describen y motivan algunas de las decisiones tomadas durante el transcurso del proyecto. Se discuten las conclusiones sobre el resultado del trabajo y las posibilidades futuras de desarrollo.

5.1 Decisiones en el trabajo

En las etapas iniciales del desarrollo del programa, se consideraron dos opciones de cómo manejar las formas geométricas, usando formas geométricas como superficies o usando nodos y diferentes tipos de conexiones para combinar.

5.1.1 Superficies y Formas Geométricas vs Nodos y Conexiones

La idea de dibujar usando superficies es usar diferentes formas geométricas simples en combinación con la superposición y otras herramientas que pueden crear geometrías casi arbitrarias. La ventaja de este enfoque es que es la forma más convencional de dibujar en otros tipos de software fácil de usar, como PowerPoint y MS Paint.

También muestra la geometría de una manera muy intuitiva. Permite una edición simple si el usuario desea cambiar la geometría con ligeras modificaciones, como mover un orificio para ver el impacto más adelante en el proceso. El uso de nodos para dibujar proporciona medios de implementación más fáciles, ya que es más similar a la estructura de Gmsh, donde todas las geometrías están definidas por puntos con algún tipo de conexión de línea. La desventaja es que el usuario tiene que saber dónde colocar todos los puntos de intersección y los bordes, lo que no siempre es la forma más intuitiva de pensar en las geometrías.

Esto también puede causar algunos problemas al querer mover solo una parte de una geometría. Además, dibujar con conexiones automáticas reduce drásticamente la cantidad de acciones necesarias para crear una geometría, ya que dibujar un polígono usando superficies requiere n + 1 acciones mientras que usar nodos y conexiones requiere 3n acciones, donde n es el número de lados en el polígono.

5.1.2 Dibujar usando solo polígonos

La idea inicial era poder dibujar utilizando cuatro formas geométricas estándar: círculo, rectángulo, elipse y polígono. Cabe destacar que el círculo es un caso especial de elipse y el rectángulo es un caso especial de polígono, lo que significa que se trata de dibujar usando líneas rectas en el polígono y líneas arqueadas para la elipse. Después de decidir usar el dibujo basado en superficies y formas geométricas, finalmente se hizo evidente que era necesario manejar las superposiciones en la geometría. Para puntos individuales, la verificación de superposiciones no fue un problema, ya que existen métodos para esto en PyQt5. Sin embargo, no había una manera fácil de verificar la superposición de elementos gráficos completos. Para manejar esto, se tuvieron que fabricar métodos manuales para exhibir el comportamiento deseado. Esto planteó un desafío tanto para los polígonos como para las elipses. Sin embargo, los métodos podrían construirse con una complejidad significativamente menor utilizando que los polígonos solo tienen líneas rectas, en comparación con las líneas arqueadas de una elipse. Esto se debe al hecho de que conocer todos los puntos se traduce automáticamente en conocer todas las líneas, ya que dos puntos siempre están conectados con las mismas líneas rectas. El problema con las líneas arqueadas es que conocer los puntos no brinda ninguna información sobre el arco de la curva y, por lo tanto, es ambiguo. Un ejemplo de esto es, si se sabe que dos puntos están contenidos dentro de un cuadrado. Si la línea es recta es equivalente a que la línea está contenida dentro del cuadrado como se ve en la Figura 5.1. Sin embargo, este no es el caso de la línea arqueada, ya que la curvatura de la línea puede hacer que se cruce con el borde del cuadrado, incluso si ambos extremos están dentro del cuadrado. Por lo tanto, durante el proyecto se decidió centrarse únicamente en implementar la estructura de los polígonos y dejar las elipses para trabajos futuros.

A partir de estos dos enfoques se decidió implementar el dibujo utilizando superficies y formas geométricas. Aunque esta alternativa se consideró la más desafiante, se supuso que sería la más fácil de usar como programa terminado.

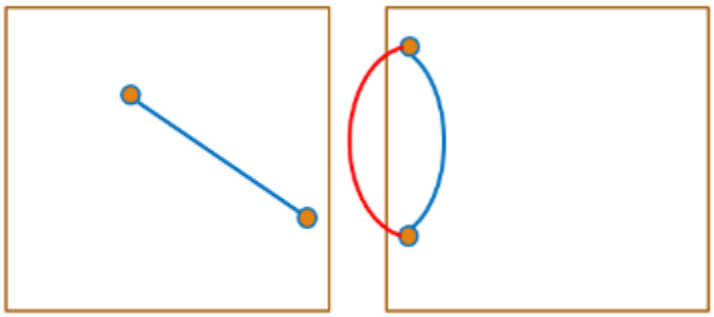

Figura 5.1: Comparación de cómo una línea recta está contenida dentro de un cuadrado donde los puntos finales están contenidos mientras que una línea arqueada está contenida dependiendo de la curvatura de la línea.

5.2 Conclusiones

El objetivo del proyecto era implementar un programa con la posibilidad de crear y editar geometrías 2D. Este objetivo se ha logrado ya que se ha desarrollado un programa y está operativo.

Sin embargo, se establecieron otros objetivos con respecto a la funcionalidad que rodea al programa. La demanda de que el programa sea fácil e intuitivo de usar, por supuesto, no es objetiva, ya que la idea sobre esto varía de persona a persona. La idea era realizar un estudio de usuarios para validar si el diseño se consideraba intuitivo o no. Desafortunadamente, debido al marco de tiempo, el programa no se terminó a tiempo para permitir un estudio de usuario que hubiera arrojado resultados útiles. En cuanto a la versatilidad de los programas, se cumplieron los objetivos en cuanto a poder lanzar el programa como un programa independiente y desde comando con opción de devolución de una geometría, también con la carga de geometrías preexistentes. Para usar como componente en otras interfaces gráficas de usuario basadas en Qt, el objetivo se cumplió parcialmente ya que la escena gráfica contiene todas las funciones principales. Se puede usar sin la ventana principal, lo que permite usarlo como un recurso con una interfaz de usuario separada. Sin embargo, algunas funcionalidades adicionales se basan en la existencia de controles y widgets disponibles en la interfaz de usuario de la ventana principal.

5.3 Trabajo futuro

Para el trabajo futuro existen muchas posibilidades de mejorar y elaborar más las funciones implementadas, así como la introducción de más funcionalidades.

5.3.1 Mejora de las funciones existentes

En la funcionalidad existente, uno de los principales inconvenientes es la baja eficiencia computacional del ajuste a los bordes. Cuando se mueve un punto, todos los bordes se recorren y, por lo tanto, se vuelve cada vez más ineficiente a medida que se agregan más bordes, lo que podría causar problemas si se crea una geometría grande o complicada. Para resolver esto, se podría pensar más en cómo hacer esto de manera más eficiente y tal vez hacer uso de operaciones matriciales, como el ajuste de punto a punto, que es mucho más eficiente. Otra opción sería explorar si se podría usar alguna otra estructura de datos con más eficiencia. En cuanto a la ausencia de un estudio de usuarios, el programa podría necesitar algunas mejoras en cuanto a la usabilidad para ser más intuitivo de usar. Además, sería muy útil averiguar qué podría mejorarse o necesitar modificaciones.

5.3.2 Funcionalidad ampliada

Desde las primeras etapas del trabajo se echa en falta la posibilidad de crear otras formas además de los polígonos, más concretamente poder crear elipses y círculos. Debido al enfoque en solo crear superficies, también se excluyó el uso de la creación de líneas. Sin embargo, agregar esto requeriría que el usuario defina manualmente qué líneas deben construir una superficie, lo que introduce una segunda forma de crear superficies. Esto combinaría la idea de dibujar utilizando superficies o nodos y conexiones, permitiendo al usuario realizar la misma acción con diferentes comandos, lo que se consideró contradictorio con la idea de mantener el programa simple e intuitivo. Otra opción sería implementar el cierre automático cuando las líneas forman una superficie, pero esto probablemente sería muy complejo de implementar y más propenso a errores del usuario ya que todas las líneas deben formar un cierre para generar una malla.

Otra herramienta útil que podría implementarse para poder crear más geometrías arbitrarias de una manera más eficiente es una función de recorte. Esto funcionaría de manera similar a la capacidad de fusión ya existente donde dos polígonos se combinan en uno, con la diferencia de que un polígono se elegiría como el que se corta en el otro. Es decir. se eliminaría el área del polígono

recortado y el solape con el polígono existente, recortando así una parte del polígono anterior. Otra forma de implementar una funcionalidad similar sería modificar la función actual de creación de agujeros para permitir la superposición en los bordes; si el agujero luego elimina el borde superpuesto, se comportaría de manera similar a un corte, ya que uno o más bordes del agujero no estarían contenidos dentro del polígono.

5.3.3 Eficiencia computacional

El problema más notable con la eficiencia computacional viene con el algoritmo de ajuste a los bordes. El ajuste a puntos podría hacerse muy eficiente ya que era posible aplicar operaciones matriciales con las coordenadas. Sin embargo, con los bordes no es posible comprobar todos los puntos de los bordes, ya que existe una gran cantidad. En cambio, las iteraciones recorren cada borde para verificar si está cerca del puntero del mouse, lo que no es tan eficiente. Como esto tiene que hacerse con cada movimiento de píxel, el aumento del tiempo de cálculo se vuelve notable y se muestra como un ligero retraso al mover el puntero y no parece tan suave. El problema aumenta con una mayor cantidad de bordes en la escena. Sin embargo, dado que no se espera que las geometrías creadas sean de una complejidad muy alta, el número de bordes debe permanecer relativamente bajo en la medida en que el retraso no cree ningún problema importante.