Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Glass3D: Generación automática de la imagen 3D del contenido de los moldes de la FCNV

Alumno: Gonzalo López Pérez

Tutor/a/es: José Vicente Álvarez Bravo, Luis Mª Fuentes García

Glass3D: Generación automática de la imagen 3D del contenido de los moldes de la FCNV

Gonzalo López Pérez

28-06-2023

Índice general

Índice de figuras

Índice de cuadros

Resumen

Este TFG surge de un proyecto de colaboración entre el Parque Científico de la Universidad de Valladolid y La Fundación Centro Nacional del Vidrio. El objetivo de dicha colaboración es la digitalización del patrimonio de moldes que usa la FCNV en la fabricación del vidrio, generando una librería 3D que contenga los moldes, obtenidos mediante el uso de un escáner, y sus piezas de vidrio correspondientes. El tamaño de esta colección es de aproximadamente 6000 moldes y la generación de las piezas de vidrio se haría mediante modelado 3D, una por una.

En esta situación es donde se aprecia la ventaja de Glass3D. Este TFG está dedicado a automatizar al máximo posible, la generación de los positivos sobre los moldes de soplado rodado, que suponen el 80 % de la colección. Dicho software es desarrollado sobre el framework Qt en C++, permite al usuario generar las piezas de vidrio en un tiempo notablemente inferior al modelado y evitando tener una formación previa sobre otro software como Blender.

Palabras claves: Qt, Qt3D, Blender, C++, 3D, Molde, STL

Parte I

Memoria del Proyecto

Capítulo 1

Descripción del proyecto

1.1. Introducción

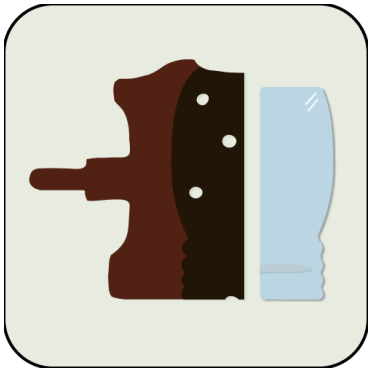

Figura 1.1: Logo de Glass3D

La motivación de realizar este TFG surge durante la participación en un proyecto de investigación de la Universidad de Valladolid, gestionado por el Parque Científico. El objetivo de dicho proyecto es digitalizar el patrimonio histórico de moldes usados por la Real Fábrica de Cristales de la Granja para la confección de piezas de vidrio.

Esta institución posee una colección de más de 6000 moldes procedentes de distintos puntos de España, de los cuales un 80 % se producen usando una técnica de soplado rodado, es decir, son simétricos. Nuestra labor durante el periodo de colaboración es la investigación y generación de positivos del mayor número de moldes posibles. El proceso habitual que llevamos a cabo es el siguiente:

- uso de un escáner de luz estructurada para generar el archivo 3D del molde,
- generación del positivo de dicho molde mediante la manipulación del archivo obtenido previamente. Para este paso utilizamos el software Blender.

En este contexto surge la motivación de este TFG: automatizar la generación del positivo de los moldes simétricos mediante un software que no requiera el uso de Blender. De esta forma, cualquier persona que continúe con el proceso de digitalización podrá hacer uso de esta aplicación de forma sencilla, además, al perder menos tiempo en la generación de un positivo aumentamos la productividad y, en consecuencia, la cantidad total de moldes procesados a entregar. La idea es que este software pueda ser utilizado a la vez que se genera el objeto 3D de un molde, mediante su escaneo.

1.2. Objetivos del trabajo

El objetivo de este trabajo es el desarrollo de una aplicación de escritorio que permita automatizar la generación de piezas de vidrio en 3D a partir de sus correspondientes moldes simétricos. Para lograrlo se realizarán las siguientes tareas:

- Formación e investigación sobre C++ y QT:
  - Formación de desarrollo en QT usando C++.
  - Familiarización con el IDE QT Creator y el desarrollo de GUIs con QWidgets.
  - Investigación de librerías para el manejo de modelos 3D (Point Cloud Library, QT3D, Assimp,...).
- Desarrollo de la aplicación que genera un positivo a partir de su molde. Con las siguientes funciones:
  - Elegir el grosor de los bordes de la pieza.
  - Definir la zona de corte de la pieza como se haría en la confección artesanal.
  - Definir la profundidad de la base interior como se haría en la confección artesanal.
- Implementación de la app en distintas plataformas de escritorio (Windows, Linux y Mac OS) para permitir su uso en el mismo momento del escaneo de moldes en la Real Fábrica de Cristales.

El uso de esta aplicación está limitado a moldes de soplado rodado, que sean simétricos. La calidad de los resultados obtenidos depende en gran parte de los parámetros que ajuste el usuario como, por ejemplo, el centro de la pieza.

1.3. Entorno de aplicación

En el entorno de Qt se pueden encontrar aplicaciones desarrolladas con objeto de visualización o interacción con modelos 3D ya generados. Hay proyectos muy útiles que permiten modificar texturas o hacer cambios de cámara animados a modo de presentación de producto en 3D.

Con QT3D nos encontramos ante una librería destinada al renderizado de escenas 3D en aplicaciones de escritorio, no ante una librería de modelado que nos permita generar o modificar polígonos dentro de un objeto 3D. Esto, sumado a la poca documentación que se ha encontrado en Internet, complicaba el desarrollo del software que se había planteado.

No obstante, al usar la versión Open Source, el código fuente es público, por lo que mediante la consulta del funcionamiento interno del framework y combinando funcionalidades de obtención de coordenadas sobre la escena, se obtiene un resultado. La generación del positivo se realiza mediante la escritura de las coordenadas de los miles de triángulos en un fichero STL usando un algoritmo.

En definitiva, este proyecto supone haber desarrollado funcionalidades de modelado 3D sobre Qt, algo que no se había hecho antes.

1.4. Entorno y tecnologías utilizadas

En este apartado se exponen las tecnologías usadas durante el desarrollo de este trabajo de fin de grado. Al ser una aplicación de escritorio se omitirá la descripción de los distintos sistemas operativos en los que es funcional (Windows, Linux, MacOS) y se situará en contexto el lenguaje de programación C++, el framework Qt y Qt3D (librerías de manejo de entidades 3D).

1.4.1. C++

Es un lenguaje de programación creado con el propósito de extender las capacidades del lenguaje de programación C para permitir el manejo de objetos, posteriormente se añadieron facilidades de programación genérica, además de los paradigmas de programación estructurada y orientada a objetos que ya existían.

Figura 1.2: Logo de C++

Características de C++:

- Sintaxis heredada del lenguaje C.
- Programación Orientada a Objetos.
- Permite la agrupación de instrucciones.
- Es portátil y tiene un gran número de compiladores en diferentes plataformas y sistemas operativos.

- Permite la separación de un programa en módulos que admiten compilación independiente.
- Es un lenguaje de alto nivel.

1.4.2. Qt

Qt es un framework multi-plataforma orientado a objetos, muy utilizado en el desarrollo de aplicaciones que requieren una interfaz gráfica. Utiliza el lenguaje de programación C++ de forma nativa, aunque puede usarse con otros lenguajes mediante bindings, como por ejemplo con Java, PHP, Pascal o Python.

Figura 1.3: Logo de Qt

Es muy usado debido a su robustez, sencillez, rendimiento nativo y compatibilidad multi-plataforma, además de sus distintas licencias, Open Source y comerciales. Actualmente existen 3 ediciones distintas de Qt:

- **GUI Framework** – edición con nivel reducido de GUI, orientado a redes y bases de datos.
- **Full Framework** – edición completa comercial
- **Open Source** – edición completa Open Source. La utilizada en este TFG.

1.4.3. Qt3D

Qt3D es un módulo incluido desde Qt 5.7 en adelante, aunque existe documentación, está incompleta. Ingenieros de la empresa KDAB se encargan del trabajo de mantenimiento, quienes se ofrecieron de forma voluntaria para hacerse cargo de dicha tarea.

Este módulo permite la interacción de escenas 3D sobre interfaces gráficas de usuario, proporcionando utilidades para renderizar en 2D y 3D, aplicar texturas, simular físicas, simular colisiones, aplicar iluminación,...etc.

Qt3D utiliza un sistema Entidad-Componente, como se puede ver en la figura 1.4. Se construyen Entidades y se les agregan Componentes; estos Componentes son los que proporcionan las características adicionales necesarias para determinar cómo se renderiza una Entidad.

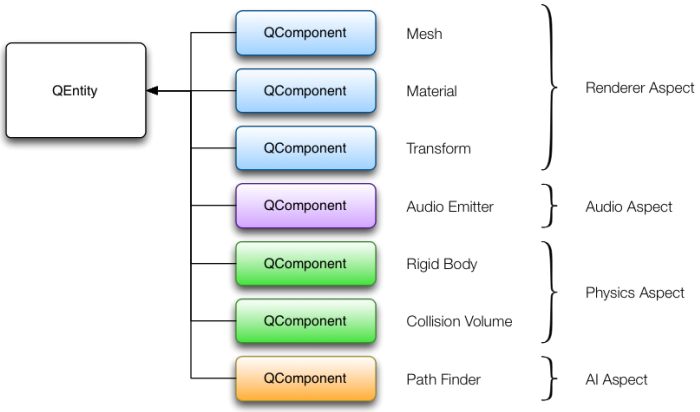

Figura 1.4: Qt3D: Entity-Component-System

1.5. Estructura de la memoria

- **Parte I:** Memoria del proyecto.
  - **Descripción del proyecto:** En este capítulo se describe brevemente la motivación del trabajo. Así como las ventajas que implica a la hora de aumentar la productividad en la digitalización del patrimonio de la Real Fábrica de Cristales de la Granja.
  - **Metodología:** Capítulo destinado a mostrar las elecciones tomadas en el proceso de desarrollo y las herramientas necesarias para llevarlo a cabo.
  - **Planificación:** Capítulo que agrupa la gestión de recursos temporales, económicos y de personal estimada al principio del desarrollo, comparando la planificación inicial con los recursos finalmente empleados.
- **Parte II:** Documentación técnica.
  - **Análisis:** Capítulo en el que se especifican los casos de uso y requisitos definidos en el desarrollo del software.
  - **Diseño:** Capítulo que agrupa las fases del diseño del software. Incluye los diagramas de clases, secuencia y la GUI.
  - **Implementación:** En este capítulo se describe, a bajo nivel, cómo se ha realizado el desarrollo de las funciones principales del proyecto y también se especifican modificaciones frente al planteamiento inicial, así como dificultades del proceso.
  - **Pruebas:** Capítulo que agrupa las pruebas realizadas sobre el software para asegurar su correcto funcionamiento e interacción con el usuario.

- **Conclusiones:** En este capítulo se realiza una breve reflexión sobre los resultados obtenidos durante el desarrollo del proyecto, así como las líneas de futuro aplicables al software.

- **Parte III:** Manuales de la aplicación.
  - **Manuales de usuario:** En esta sección se agrupan los dos manuales incluidos en la ayuda de la aplicación: uso básico y avanzado.

Capítulo 2

Metodología

2.1. Proceso de desarrollo

Para el desarrollo de este proyecto se ha utilizado un modelo de desarrollo iterativo. No se ha usado un modelo incremental debido a que se tenía muy claro el resultado esperado por el cliente y no se podían obtener productos funcionales en cada iteración, pues las primeras iteraciones están destinadas a la formación del lenguaje e investigación de librerías. Al ser un equipo de trabajo reducido, tampoco se ha abordado el uso de metodologías ágiles.

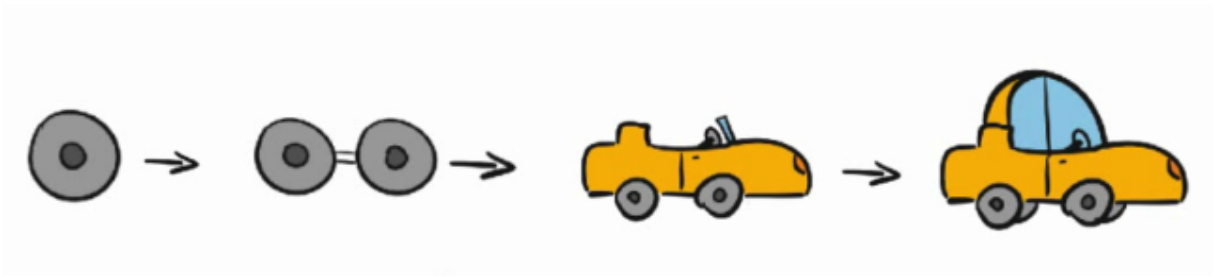

Figura 2.1: Modelo de desarrollo iterativo

Al trabajar sobre el framework Qt se usa un paradigma de programación orientada a objetos, trabajando este, de forma nativa, con el lenguaje de programación C++.

2.2. Herramientas utilizadas

Entorno de trabajo

1. El proyecto se ha desarrollado conjuntamente entre un equipo de escritorio y un portátil. En las primeras pruebas se usó Qt en Linux por la facilidad a la hora de integrar librerías Open Source. Al tomar la decisión de usar la librería nativa Qt3D se define el entorno de trabajo en Windows 11, por comodidad de uso.

Lenguaje de programación

1. El lenguaje de programación utilizado ha sido C++, trabajando de forma nativa sobre el framework Qt en su versión 6.5.0, aunque existía la posibilidad de combinar el uso de C++ con QML para el módulo de Qt3D. Se opta por mantener C++ con el objetivo de no invertir tiempo extra en la formación de QML.

2. El IDE utilizado ha sido Qt Creator en su versión 10.0.0, desarrollado por Qt Company, empresa propietaria de Qt, para facilitar el uso y aprendizaje de sus propias librerías.

Documentación

1. La memoria se ha realizado a partir de la plantilla para TFGs de LaTeX proporcionada por la Escuela de Ingeniería Informática de Segovia. Usando el editor online *OverLeaf* [8].

2. Para la creación de los diagramas incluidos en esta memoria se ha utilizado la web *draw.io*[6].

3. La edición de imágenes se ha llevado a cabo con la web *Canva*[2].

4. La herramienta usada para crear los gráficos de barras ha sido Microsoft Excel de Office 365.

5. El logo de la aplicación se ha diseñado usando Adobe Illustrator.

Comunicación

La comunicación con los tutores se ha realizado mayormente de forma presencial durante las jornadas en la Real Fábrica de Cristales de la Granja, usando además Microsoft Teams y el correo electrónico de la UVa como opciones adicionales comunicación y revisión de documentación.

2.3. Definición de siglas y abreviaturas

Abreviatura	**Significado**
FCNV	Fundación Centro Nacional del Vidrio
STL	Standard Triangle Language
GUI	Graphic User Interface
PCL	Point Cloud Library

Cuadro 2.1: Definición de siglas y abreviaturas

Capítulo 3

Planificación

En este capítulo se detalla la distribución de recursos prevista en las distintas iteraciones del proyecto para cumplir los objetivos y requisitos propuestos, así como los posibles riesgos que puedan retrasar el desarrollo.

3.1. Planificación temporal

3.1.1. Planificación inicial

En esta sección se abordará la distribución temporal al inicio del desarrollo del proyecto para luego contrastar con el resultado final y comprobar con qué exactitud se cumple esta previsión.

Distribución temporal

El proyecto comienza a desarrollarse el día 7 de Marzo del 2023 y se estima que finalice el 23 de Junio del 2023. Para ello se propone invertir 4 horas diarias de media en los días laborables de este periodo. Entre las fechas en las que se desarrolla tenemos un total de 76 días laborables, por lo que resulta una estimación de **304 horas totales**.

El reparto de horas se ha hecho basándose en otros proyectos desarrollados desde una situación de partida similar, con mucha labor inicial de formación e investigación, teniendo en cuenta también la probabilidad y gravedad de riesgos asociados a cada iteración.

El reparto temporal de las iteraciones será el siguiente:

- **Iteración 1 (46 horas):** Esta iteración tiene como primer objetivo realizar una formación del lenguaje de programación C++ y su aplicación en el uso del framework Qt. Además tiene como segundo objetivo la investigación de librerías de terceros propuestas por los tutores sobre el manejo de entidades 3D en Qt; este segundo objetivo también comprende desarrollar ejemplos y comprobar las posibilidades que ofrece cada librería para así asegurarnos no tener que cambiarla en mitad del proyecto. En el momento de inicio del proyecto se van a comprobar las librerías PCL (Point Cloud Library) y Assimp.

- **Iteración 2 (152 horas):** El objetivo principal de esta iteración es desarrollar las principales funcionalidades de manejo de entidades 3D que permitan generar los resultados esperados por la Real Fábrica de Cristales de la Granja. Dividiendo este objetivo en varios más específicos el primero será conseguir registrar el contorno de un molde y el segundo será generar una función que simule una revolución a partir del contorno para construir el positivo del molde. Además de las funcionalidades especificadas se espera que aparezcan más requisitos de funciones complejas durante el desarrollo.
- **Iteración 3 (46 horas):** El objetivo de esta iteración es desarrollar una interfaz gráfica amigable que haga uso de las funciones de manejo 3D de la iteración anterior. Durante esta iteración se irán generando prototipos funcionales desde casos específicos a más generales.
- **Iteración 4 (60 horas):** La iteración final consta de tres objetivos: El primero es la mejora y corrección de errores del software desarrollado; el segundo es realizar las correspondientes pruebas para comprobar el correcto funcionamiento, y el tercero la confección de los manuales de instalación y funcionamiento.

En la figura 3.1 se puede ver un gráfico de la distribución temporal y la acumulación total de horas de trabajo a lo largo de las iteraciones.

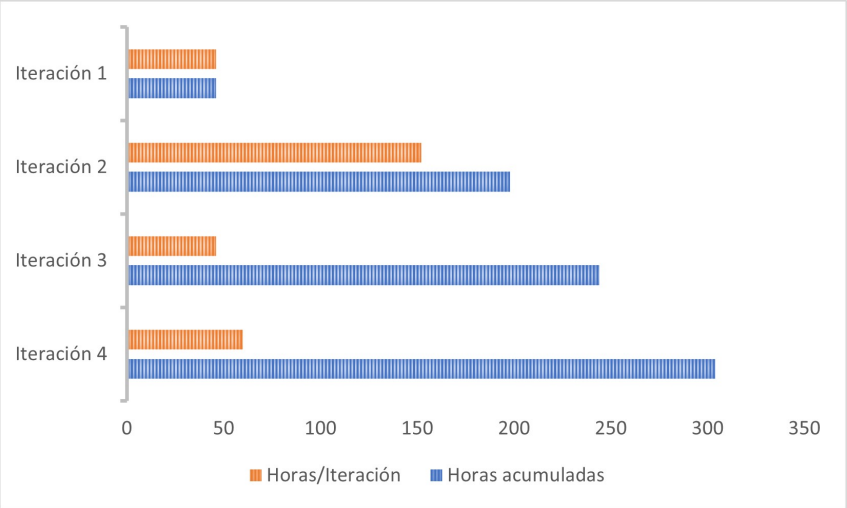

Figura 3.1: Distribución temporal inicial

Análisis de riesgos

Durante el desarrollo del proyecto pueden darse situaciones que afecten a la planificación del mismo, afectando en forma de retrasos temporales que se convertirán en incrementos de las cantidades presupuestadas para los recursos. Por ello, es conveniente tener identificados los riesgos con mayor probabilidad o impacto sobre la planificación. En la tabla 3.1 se muestran los riesgos a los que está expuesto el desarrollo de este proyecto. En base a estos riesgos se calcula la exposición de cada uno, multiplicando las horas de retraso estimadas por la probabilidad. Además se incluyen planes de reducción y mitigación para cada riesgo, que se pueden ver en la tabla 3.2.

ID	Riesgo	Probabilidad	Descripción	Retraso
1	Falta de experiencia en C++ y Qt	0.1	Invertir más tiempo en alguna fase del desarrollo por falta de experiencia.	15
2	Incompatibilidad de versiones de Qt	0.1	Invertir tiempo extra en ajustar código a las nuevas versiones de Qt.	8
3	Elección incorrecta de la librería de manejo 3D	0.15	La librería elegida en la primera iteración no cumple los requisitos para realizar el proyecto.	50
4	Desarrollo inviable de una funcionalidad 3D	0.3	No es posible desarrollar una funcionalidad planteada como se haría en Blender.	30
5	Planificación incorrecta	0.05	El desarrollo no se ajusta a la planificación inicial.	12
6	Pérdida de datos	0.05	Perder cambios realizados sobre el software.	4
7	Añadir/Modificar funcionalidades	0.2	Cambio de requisitos o ajustes en la GUI.	15
8	Problemas con el equipo de desarrollo	0.05	El equipo de desarrollo no es utilizable.	10

Cuadro 3.1: Análisis de riesgos

ID	Riesgo	Exposición	Plan de reducción	Plan de mitigación
1	Falta de experiencia en C++ y Qt	1.5	Documentarse bien antes de comenzar desarrollos con aspectos no tratados en la formación.	Invertir más horas de las necesarias inicialmente.
2	Incompatibilidad de versiones de Qt	0.8	Evitar la reutilización de código de versiones anteriores de Qt.	Comparar la diferencia de versiones sobre la documentación oficial y corregir el código.
3	Elección incorrecta de la librería de manejo 3D	7.5	Realizar la investigación inicial de las librerías comprobando que se cumplan una mayoría de las funcionalidades necesitadas.	Importar sobre el proyecto librerías de terceros que cumplan los requisitos necesarios.
4	Desarrollo inviable de una funcionalidad 3D	9	Comprobar exhaustivamente que una idea sobre el desarrollo es aplicable en el contexto de la librería de manejo 3D escogida.	Invertir tiempo extra pensando la forma de llegar al resultado esperado de otra manera.
5	Planificación incorrecta	0.6	Extender ligeramente la duración estimada de las iteraciones para disponer de tiempo extra.	Intentar reducir tiempos en futuras tareas que puedan requerir menos esfuerzo.
6	Pérdida de datos	0.2	Sincronizar la información con la cuenta de OneDrive Universitaria.	Recuperar una copia local de otro dispositivo o usar el histórico de OneDrive.
7	Añadir/Modificar funcionalidades	0.3	Realizar una planificación lo más precisa posible para no tener que realizar muchos cambios durante el desarrollo.	Comunicarse con el cliente y dejar por escrito las modificaciones solicitadas.
8	Problemas con el equipo de desarrollo	0.5	Disponer de más de un equipo de desarrollo.	Utilizar el equipo de desarrollo auxiliar mientras se solucionan los problemas con el principal.

Cuadro 3.2: Análisis de riesgos: reducción y mitigación

3.1.2. Planificación final

En esta sección realiza una comparación de las horas planificadas inicialmente para cada iteración con las reales, que han sido necesarias para completar el desarrollo. En la figura 3.2 se compara mediante gráficos de barras la diferencia de tiempos.

- **Iteración 1 (50 horas):** Durante esta iteración se completo la formación sobre C++ y Qt, además se realizaron investigaciones a conciencia sobre las librerías planificadas inicialmente. Finalmente, la librería elegida como encargada del manejo 3D fue Qt3D, incluida de forma nativa en Qt y usada para desarrollar un prototipo de la aplicación, que permitía importar modelo 3D para visualizarlo e interactuar con él de forma sencilla. El retraso de esta iteración frente a la planificación inicial se debe a que se ha hecho efectivo el riesgo 2 (Incompatibilidad de versiones de Qt) durante el periodo formativo.

- **Iteración 2 (145 horas):** Aunque durante esta iteración se ha hecho efectivo el riesgo 4 (Desarrollo inviable de una funcionalidad 3D), la reducción de horas sobre la planificación inicial de esta iteración compensa las horas extra invertidas en la anterior. Durante esta iteración se han desarrollado las principales funciones de manejo 3D requeridas para obtener los resultados esperados posteriormente. Al ser funciones complejas se han desarrollado con casos de ejemplo específicos sobre el código, sin abordar la interacción con el usuario.

- **Iteración 3 (42 horas):** En esta iteración se vuelve a requerir un tiempo menor al esperado en el desarrollo de la GUI. Durante la iteración se ha diseñado la interfaz de la aplicación, tratando de ser lo más intuitiva posible y sirviendo de control para las funciones desarrolladas en la iteración anterior.

- **Iteración 4 (55 horas):** A lo largo de esta iteración se han corregido errores, desarrollado mejoras en la GUI, realizado pruebas y documentado el proyecto.

La suma de estas iteraciones nos deja un total de **292 horas** reales acumuladas durante el desarrollo del proyecto, lo que supone 12 horas menos a lo planificado inicialmente. Al repartirse las jornadas en 4 horas diarias, se ha finalizado el proyecto con **3 días de antelación**.

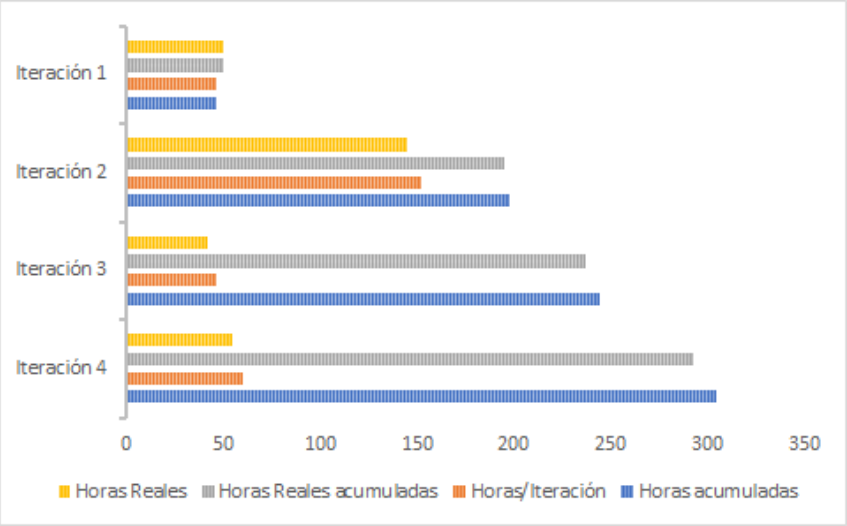

Figura 3.2: Distribución temporal final

3.2. Presupuesto económico

En este apartado se realiza el cálculo de los costes estimados para los recursos necesarios en el proyecto. Se especifica el coste de los recursos Hardware y Software, así como de los salarios estimados para los desarrolladores.

3.2.1. Hardware y software

Sobre el software no se percibe ningún gasto, pues se ha desarrollado sobre la versión OpenSource de Qt y las herramientas adicionales utilizadas son gratuitas. El único paquete de software que sería de pago es Microsoft Office 365 pero todo lo utilizado está incluido en la licencia que proporciona el correo universitario.
En cambio, en el hardware, sí se perciben gastos. Para la realización del proyecto se han usado dos equipos de trabajo, un pc de sobremesa y un portátil. En la tabla 3.3 se muestra el presupuesto hardware total.

- **Equipo de sobremesa:** El coste de este equipo fue de 892 € y la vida útil de un equipo de sobremesa se estima en 6 años. Por lo tanto, haciendo un cálculo sencillo se puede obtener el gasto que tendrá el equipo durante las 304 horas que dura el proyecto. $\frac{304}{52560} * 892 = 5{,}15$ €

- **Equipo portátil:** De la misma forma que para el equipo de sobremesa se calcula el coste estimado del portátil. Teniendo en cuenta que su coste fue de 800 € y su vida útil estimada es de 4 años obtenemos el siguiente cálculo. $\frac{304}{35040} * 800 = 6{,}94$ €

Dispositivo	Coste
Equipo de sobremesa	5,15 €
Equipo portátil	6,94 €
Coste total	12,09 €

Cuadro 3.3: Presupuesto hardware

3.2.2. Personal

En este apartado se explicará el presupuesto salarial del único desarrollador del proyecto. Según la web *glassdoor* el salario medio de un desarrollador C++ es de 33.000 € anuales. Teniendo en cuenta que se trabajan 4 horas diarias, el salario será de una media jornada sobre el comentado anteriormente, 16500 € anuales. Entonces, al ser la duración del proyecto de 3,5 meses se puede obtener fácilmente la estimación salarial a partir del siguiente cálculo. $\frac{16500}{12} * 3{,}5 = 4812{,}5$ €

3.2.3. Presupuesto total

En este apartado se calcula el presupuesto total estimado para el desarrollo del proyecto. Sumado a los apartados anteriores se deben añadir gastos estimados en términos de red eléctrica o conexión a Internet.

- **Red eléctrica:** Según la web *tarifaluzhora.es*, a día 07/03/2023 el precio medio de luz es de 0.1975 €/kwh. Usando de referencia el máximo consumo del ordenador de sobremesa, que no superaría los 500W a máximo rendimiento, se reduce este precio por hora a la mitad. Entonces, multiplicando por las horas del proyecto, el coste de consumo eléctrico del proyecto será el siguiente. $\frac{0{,}1975}{2} * 304 = 30{,}02$ €
- **Conexión a Internet:** La tarifa de Internet usada durante el desarrollo tiene un coste de 50 € al mes, multiplicado por los 3,5 meses de duración del proyecto resulta en 175 € que se suman al presupuesto total.

Finalmente, en la tabla 3.4 se puede ver resumidos el valor de los gastos tratados en esta sección. Así como el total del presupuesto económico del proyecto, **5029,79 €.**

Recurso	**Coste**
Hardware	12,09 €
Personal	4812,5 €
Red eléctrica	30,02 €
Conexión a Internet	175 €
Coste total	**5029,79** €

Cuadro 3.4: Presupuesto total

Parte II

Documentación técnica

Capítulo 4

Análisis

4.1. Requisitos

4.1.1. Iteración 1:

El objetivo de la primera iteración es obtener una buena base en el desarrollo de aplicaciones con el lenguaje C++, familiarizarse con las ventajas y paquetes más importantes del framework Qt y elegir una librería que permita el manejo de entidades 3D. En términos de desarrollo se obtendrá una versión muy básica de la aplicación que permita la importación de un objeto 3D. Los requisitos funcionales y no funcionales se pueden consultar en las tablas 4.1 y 4.2 respectivamente.

ID	Nombre	Descripción
RF_01	Crear escena 3D	El sistema debe crear una escena 3D sobre la interfaz gráfica que permita al usuario interactuar con las entidades.
RF_02	Importar objeto 3D	El sistema debe permitir al usuario la importación de un modelo 3D que se muestre en la interfaz gráfica.
RF_03	Mover cámara	El sistema debe permitir al usuario modificar el movimiento de la cámara para revisar los modelos desde distintos ángulos.

Cuadro 4.1: Requisitos funcionales de la iteración 1

ID	Nombre	Descripción
RNF_01	Entorno de desarrollo	El sistema debe desarrollarse con el lenguaje C++.
RNF_02	Movimiento de cámara	El movimiento de la cámara debe ser lo más intuitivo posible para el usuario y permitir una velocidad de desplazamiento equilibrada.
RNF_03	Formato de importación	La importación de objetos 3D debe ser válida para ficheros STL, tanto en formato binario como ASCII.

Cuadro 4.2: Requisitos no funcionales de la iteración 1

4.1.2. Iteración 2:

Esta iteración es la de mayor complejidad técnica, por ello se estima una duración de más horas respecto a las otras tres. Consta de varios objetivos al tratarse del desarrollo de varias funcionalidades, como por ejemplo la selección de puntos sobre el modelo 3D importado o la propia generación del positivo, en base a los valores introducidos posteriormente por el usuario. Varios de los requisitos descritos para esta iteración se usarán en conjunto en el flujo de trabajo final. Estos requisitos se describen en las tablas 4.3 y 4.4.

ID	Nombre	Descripción
RF_04	Definir centro	El sistema debe permitir al usuario definir el centro de un molde al seleccionar un punto del contorno.
RF_05	Generar preview	El sistema debe permitir al usuario generar una preview escaneando la mitad del contorno de un modelo 3D desde un centro definido. Permitiendo elegir si se escanea la mitad izquierda o derecha.
RF_06	Generar positivo	El sistema debe permitir al usuario generar un positivo del modelo 3D cargado a partir del contorno escaneado y los valores de precisión especificados por el usuario.

Cuadro 4.3: Requisitos funcionales de la iteración 2

ID	Nombre	Descripción
RNF_04	Seleccionar puntos	El sistema debe obtener las coordenadas de un punto seleccionado por el usuario en el modelo 3D.
RNF_05	Detectar colisión	El sistema debe ser capaz de detectar las coordenadas de una colisión desde un punto X del que sale un rayo de luz en dirección a un punto Y.
RNF_06	Detección de colisiones múltiples	El sistema debe permitir que se ejecuten múltiples detecciones de colisiones en el menor tiempo posible y sin omitir ninguna dirección especificada.
RNF_07	Definir centro	El sistema definirá el centro de la forma más óptima posible para evitar múltiples acciones del usuario.
RNF_08	Generar Positivo	El sistema generará el positivo del molde en formato STL.

Cuadro 4.4: Requisitos no funcionales de la iteración 2

4.1.3. Iteración 3:

El objetivo de esta iteración es la creación de la interfaz gráfica que hará uso de todas las funcionalidades desarrolladas hasta el momento. El funcionamiento se dividirá en 3 estados por los que irá pasando la escena.

1. **Definir centro:** El usuario deberá seleccionar un punto en el contorno del molde que, mediante la detección de colisiones en direcciones opuestas, generará un centro de escena desde el que trabajar en los siguientes pasos.

2. **Generar preview:** Desde esta etapa del proceso se podrá definir en qué dirección se va a realizar el escaneo del contorno, izquierda o derecha. Una vez escaneado el contorno se generarán dos planos, uno en la parte inferior y otro en la superior del contorno escaneado. El usuario puede desplazar estos dos planos para definir desde donde se generará la base interior del positivo y dónde se realizará el corte superior de la pieza. También puede definir el valor en milímetros del grosor de la pieza.

3. **Generar positivo:** En esta última etapa se muestra el positivo generado a partir de los valores definidos en el paso anterior. También permite ver el archivo del molde para comprobar si la generación se ha realizado de la forma esperada.

Los requisitos de esta iteración se detallan en las tablas 4.5 y 4.6.

ID	Nombre	Descripción
RF_07	Seleccionar fichero	El sistema permitirá al usuario seleccionar un fichero stl a importar en cualquier momento.
RF_08	Reiniciar vista	El sistema permitirá al usuario restablecer la posición original de la cámara de forma sencilla.
RF_09	Ajustes	El sistema permitirá al usuario modificar valores por defecto de la aplicación o la precisión de la generación mediante una ventana de ajustes.
RF_10	Generar centro	El sistema informará al usuario de las coordenadas del centro generado, además de generar una esfera roja en dicha posición a forma de referencia visual.
RF_11	Definir dirección del escaneo	El sistema permitirá al usuario definir en qué dirección quiere hacer el escaneo del contorno. Izquierda o derecha.
RF_12	Definir grosor	El sistema debe permitir al usuario modificar el valor que especifica el grosor del borde para el positivo a generar.
RF_13	Mover planos	El sistema debe permitir al usuario desplazar los planos generados después de escanear el contorno.
RF_14	Revisar resultado	El sistema permitirá al usuario alternar entre mostrar el positivo, su molde o ambos para facilitar la revisión del resuldado.
RF_15	Guardar resultado	El sistema permitirá al usuario elegir una ruta donde guardar el positivo generado durante el flujo de trabajo.

Cuadro 4.5: Requisitos funcionales de la iteración 3

ID	Nombre	Descripción
RNF_09	Seleccionar fichero	Al seleccionar un fichero se reiniciará el flujo de trabajo volviendo al paso 1 para comenzar una nueva generación.
RNF_10	Ajustes	La modificación de ajustes solo se permitirá desde el paso 1.
RNF_11	Mensajes de error	Los errores en la selección de puntos o las acciones no permitidas para el usuario se mostrarán en pantalla mediante una ventana de información.
RNF_12	Movimiento entre fases	El cambio del paso sobre el que trabaja el usuario estará limitado a los botones correspondientes para avanzar de fase. No permitiendo una navegación libre o saltarse un paso.
RNF_13	Barras de progreso	El estado de las funciones de generación de preview y positivo se mostrará en pantalla mediante una barra de progreso.

Cuadro 4.6: Requisitos no funcionales de la iteración 3

4.1.4. Iteración 4:

Los objetivos de esta iteración son los siguientes:

- Desarrollar mejoras y corregir errores en la aplicación.
- Realizar todas las pruebas necesarias para certificar el buen funcionamiento de la aplicación.
- Diseñar e implementar dentro de la aplicación ayudas en forma de manuales.

Los requisitos de la iteración final se detallan en las tablas 4.7 y 4.8.

ID	Nombre	Descripción
RF_16	Manuales	El sistema debe permitir al usuario acceder a los manuales del software desde una sección de ayuda en la aplicación.

Cuadro 4.7: Requisitos funcionales de la iteración 4

ID	Nombre	Descripción
RNF_14	Evitar sobrecarga de entidades	El flujo de trabajo de la aplicación debe usar el número de entidades 3D mínimo indispensable para evitar sobrecargas.
RNF_15	Optimización	La ejecución de las funciones de mayor duración deben reducirse el máximo posible.
RNF_16	Evitar errores	Aunque el usuario defina los planos fuera del molde 3D el positivo se genera igualmente sin errores.
RNF_17	Pruebas	La aplicación se someterá a todas las pruebas que se precisen para asegurar su buen funcionamiento.

Cuadro 4.8: Requisitos no funcionales de la iteración 4

4.2. Casos de uso

En esta sección se especifican los casos de uso requeridos para lograr el funcionamiento deseado por el usuario.

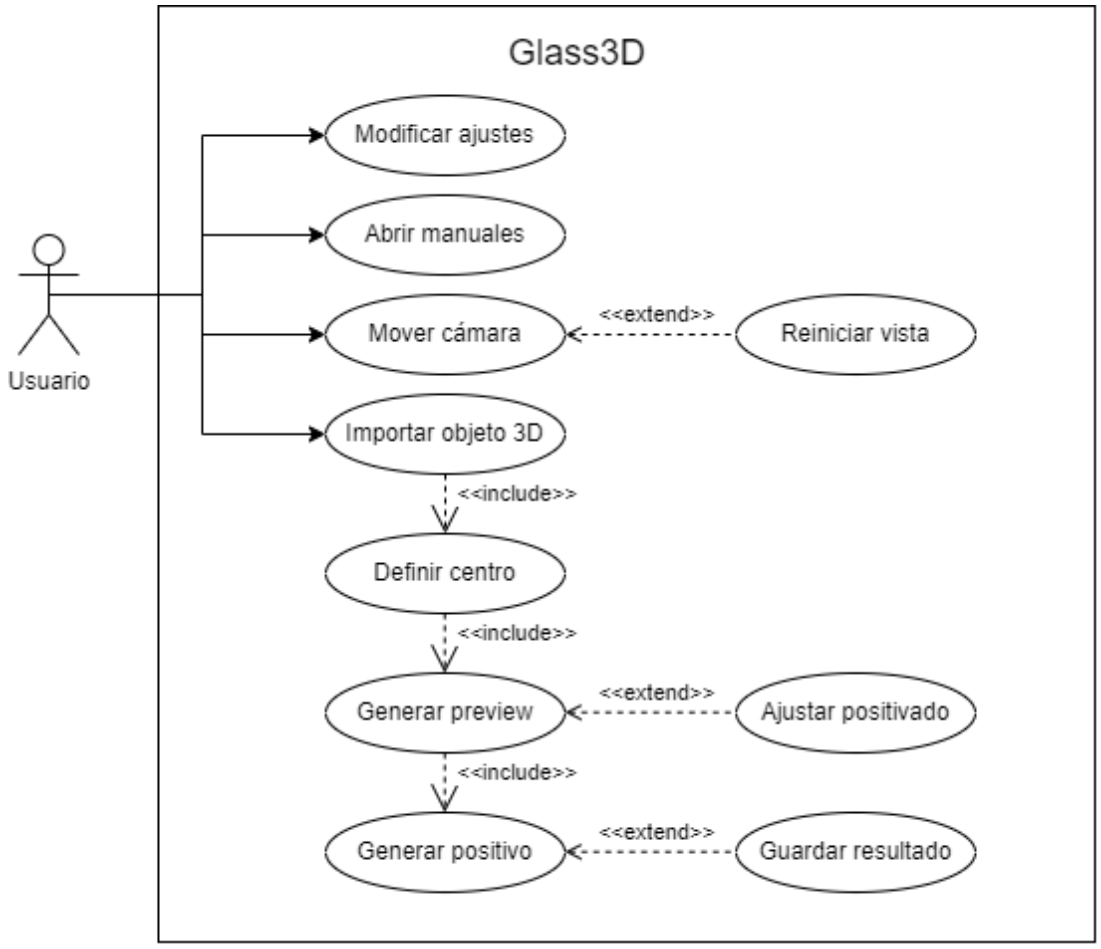

Figura 4.1: Diagrama de casos de uso

Nombre e ID del CU	CU-01. Modificar ajustes
Actor	Usuario
Descripción	El actor abre una ventana de modificación de los ajustes de la aplicación.
Postcondiciones	POST-1. La configuración establecida se guardará en un fichero INI en la ruta del programa.

Flujo normal	FN1 El actor abre la ventana de modificación de ajustes. FN2 El actor modifica los valores que precise y acepta los cambios. FN3 El sistema guarda estas modificaciones en el archivo INI de configuración. FN4 El sistema aplica los valores definidos en los ajustes.
Flujo alternativo 1	FA2 Si el actor cancela la modificación se cierra la ventana sin guardar.
Prioridad	Baja

Cuadro 4.9: CU-01. Modificar ajustes

Nombre e ID del CU	CU-02. Importar modelo 3D
Actor	Usuario
Descripción	El actor carga un nuevo modelo 3D sobre la aplicación seleccionando la ruta de un fichero STL.
Postcondiciones	POST-1. Al importar un nuevo modelo se borran todos los datos del modelo anterior, reiniciando el flujo de trabajo de la aplicación.
Flujo normal	FN1 El actor selecciona la ruta del fichero STL a cargar. FN2 El sistema intenta cargar los datos de geometría del archivo en memoria. FN3 Si no se producen fallos, el modelo 3D se muestra en la escena y se cargan los paneles para continuar el flujo de trabajo.
Flujo alternativo 1	FA1 Si el usuario cancela la selección del fichero a cargar, no sucede nada.
Prioridad	Alta

Cuadro 4.10: CU-02. Importar modelo 3D

Nombre e ID del CU	CU-03. Mover cámara
Actor	Usuario

Descripción	Una vez se ha importado un modelo 3D el actor desplaza la cámara de la escena haciendo uso del ratón.
Precondiciones	PRE-1. Se debe haber cargado un modelo 3D sobre la escena.
Flujo normal	FN1 El actor hace clic izquierdo sobre la escena y desplaza el cursor simultáneamente. FN2 El sistema calcula el movimiento que debe realizar la cámara en función de la dirección de movimiento del cursos y una velocidad predefinida. FN3 La cámara se desplaza de forma prácticamente inmediata al movimiento del cursor.
Prioridad	Media
Otra info	La posición inicial de la cámara se podrá reiniciar fácilmente mediante otra acción del actor, definida en el CU-04

Cuadro 4.11: CU-03. Mover cámara

Nombre e ID del CU	CU-04. Reiniciar vista
Actor	Usuario
Descripción	El actor reinicia la posición inicial de la cámara.
Precondiciones	PRE-1. Se debe haber cargado un modelo 3D sobre la escena.
Flujo normal	FN1 El actor selecciona la opción de reiniciar vista. FN2 El sistema devuelve la cámara a su posición y orientación inicial.
Prioridad	Baja

Cuadro 4.12: CU-04. Reiniciar vista

Nombre e ID del CU	CU-05. Definir centro
Actor	Usuario
Descripción	El actor selecciona un punto sobre el contorno del molde 3D importado para definir a qué profundidad creará el sistema el nuevo centro de la escena.
Precondiciones	PRE-1. Se debe haber cargado un modelo 3D sobre la escena.

Flujo normal	FN1 El actor selecciona un punto del modelo haciendo clic derecho. FN2 El sistema realiza una detección de colisiones en direcciones contrarias del eje X. FN3 Si el sistema obtiene dos colisiones calcula la distancia entre ambas y define el centro en el punto medio.
Flujo alternativo 1	FA3 Si el sistema detecta menos de dos colisiones muestra un mensaje informativo para que el actor lo intente, seleccionando otro punto del modelo.
Prioridad	Alta

Cuadro 4.13: CU-05. Definir centro

Nombre e ID del CU	CU-06. Generar preview
Actor	Usuario
Descripción	El actor inicia la generación de la preview, que ejecuta el escaneo del contorno del molde y crea los planos de ajuste.
Precondiciones	PRE-1. Se debe haber definido un centro usando el CU-05.
Flujo normal	FN1 El actor inicia la generación de la preview. FN2 El sistema realiza un escaneo del contorno en la dirección establecida (izquierda o derecha). FN3 Al terminar el escaneo el sistema dibuja una línea que une todos los puntos detectados, además de los planos de corte para el positivado.
Flujo alternativo 1	FA1 El actor modifica la dirección del escaneo y posteriormente inicia el flujo normal.
Prioridad	Alta
Otra info	La duración del proceso de escaneo dependerá del número de puntos a detectar que el actor haya configurado en la ventana de ajustes y del hardware del equipo.

Cuadro 4.14: CU-06. Generar preview

Nombre e ID del CU	CU-07. Ajustar positivado

Actor	Usuario
Descripción	El actor modifica los valores que se usarán en la generación del positivo, como el grosor, la altura del plano de corte superior y la altura de del plano para generación de la base interior.
Precondiciones	PRE-1. Se debe haber generado la preview.
Flujo normal	FN1 El actor modifica los valores decimales que precise. FN2 En caso de modificarse los valores de altura de los planos, el sistema desplaza los planos en la preview para una mejor visualización.
Flujo alternativo 1	FA1 El actor no modifica ningún valor por defecto.
Prioridad	Media

Cuadro 4.15: CU-07. Ajustar positivado

Nombre e ID del CU	CU-08. Generar positivo
Actor	Usuario
Descripción	El actor inicia la generación del positivo final.
Precondiciones	PRE-1. Se debe haber generado la preview.
Flujo normal	FN1 El actor inicia la generación del positivo. FN2 El sistema comienza la generación mediante una rotación de los puntos sobre el contorno escaneado anteriormente. FN3 El sistema muestra el resultado en la escena, permitiendo ocultar o mostrar tanto el positivo como el molde.
Prioridad	Alta

Cuadro 4.16: CU-08. Generar positivo

Nombre e ID del CU	CU-09. Guardar resultado
Actor	Usuario
Descripción	El actor revisa la generación del positivo y guarda el resultado.
Precondiciones	PRE-1. Se debe haber generado el positivo.

Flujo normal	FN1 El actor está conforme con el resultado y comienza el proceso de guardado y selecciona la ruta de destino. FN2 El sistema guarda el resultado en la ruta indicada.
Flujo alternativo 1	FA1 El usuario no está conforme con el resultado obtenido y vuelve al paso anterior para ajustar de nuevo el positivado.
Prioridad	Alta

Cuadro 4.17: CU-09. Guardar resultado

Nombre e ID del CU	CU-10. Abrir manuales
Actor	Usuario
Descripción	El actor utiliza la opción de ayuda para abrir los manuales de la aplicación.
Flujo normal	FN1 El actor selecciona un manual en la sección de ayuda. FN2 El sistema envía una orden al equipo para abrir el manual seleccionado, con el programa que se use por defecto para documentos PDF.
Prioridad	Baja

Cuadro 4.18: CU-10. Abrir manuales

Capítulo 5

Diseño

En este capítulo se abordan las labores de diseño de la aplicación. Se incluyen los diagramas de clases y secuencia.

5.1. Diagrama de clases

En la figura 5.1 se muestra un diagrama de clases simplificado, las clases completas se muestran en la sección de diagramas A.1 de los apéndices. A continuación se hablará brevemente del uso que tiene cada clase en el funcionamiento de la aplicación.

- **MainWindow:** Esta clase controla la interfaz gráfica principal de la aplicación. Se encarga de gestionar las acciones del usuario y transmitir las órdenes precisas a la clase encargada de la gestión de modelos 3D.

- **Settings:** Al igual que la clase anterior, esta gestiona la interfaz gráfica de la ventana de ajustes de Glass3D y guarda las modificaciones especificadas por el usuario en el fichero de configuración INI.

- **Custom3DWindow:** Es la clase principal de la aplicación. Hereda de la clase Qt3DExtras::Qt3DWindow y se encarga de todo el manejo de las funcionalidades 3D. De forma nativa, gracias a sus funcionalidades heredadas, crea la escena 3D e importa los modelos seleccionados por el usuario. Para el resto de sus tareas de más complejidad se apoya en otras clases descritas a continuación.

- **CustomCameraController:** Esta clase se conecta con la cámara que implementa la escena 3D, dotándola de una movilidad más fluida e intuitiva que la proporcionada por la clase nativa Qt3DExtras::QOrbitCameraController. Esta clase ha sido desarrollada siguiendo un tutorial [9], realizando algunos ajustes e implementando la función para hacer zoom con la rueda del ratón.

- **ConsecutiveRayCaster:** Esta clase es la encargada de emitir rayos de luz y devolver los puntos de colisión obtenidos. Internamente hace uso también de la clase

RayCastHandler. Ambas clases se han obtenido mediante modificaciones sobre el ejemplo "simple-cpp" de la empresa KDAB, encargada del mantenimiento de Qt3D; se usan para calcular el centro del modelo y escanear su contorno desde Custom3DWindow.

- **Mesh:** La clase Mesh, junto a las clases **Triangles** y **Vec3**, se usa para realizar la exportación del positivo generado en formato STL ASCII. Estas clases han sido obtenidas desde un proyecto de GitHub llamado "stl-creator" [7] añadiendo la funcionalidad de exportar en formato STL binario.

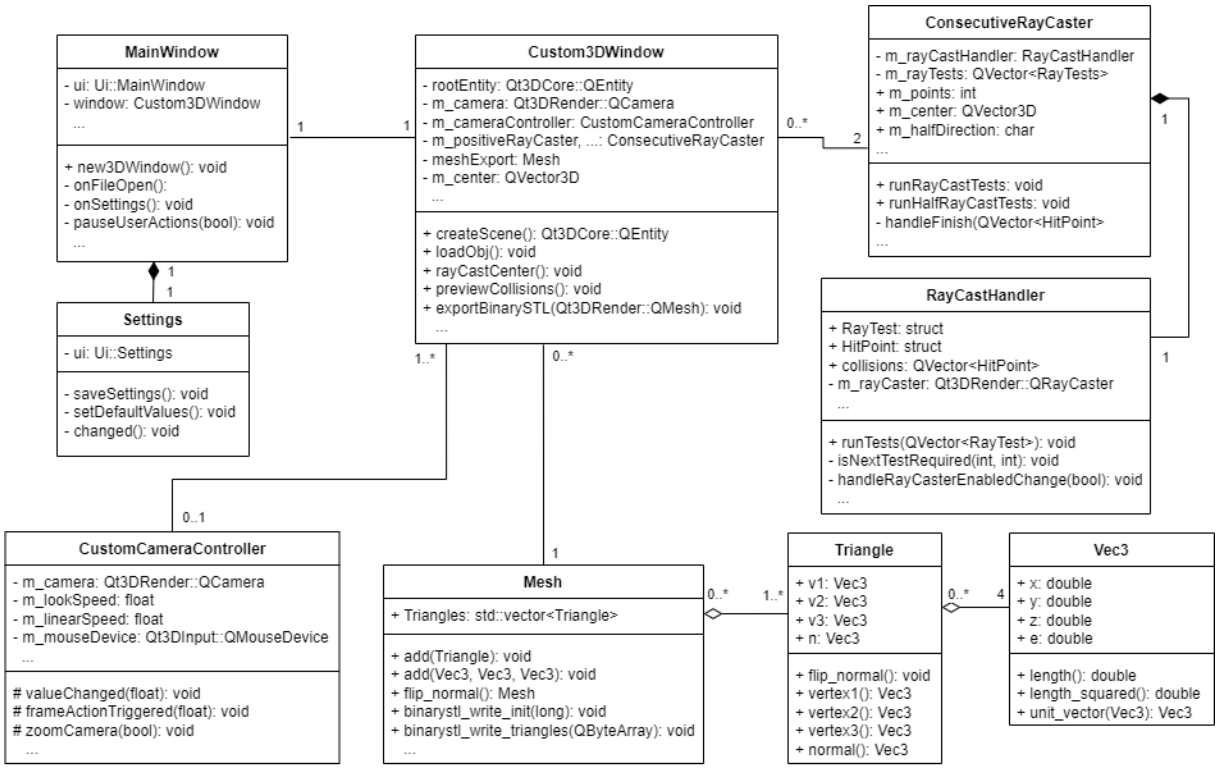

Figura 5.1: Diagrama de clases simplificado

5.2. Diagramas de secuencia

En esta sección se muestran los diagramas de secuencia para cada caso de uso expuesto en la sección 4.2: Casos de Uso.

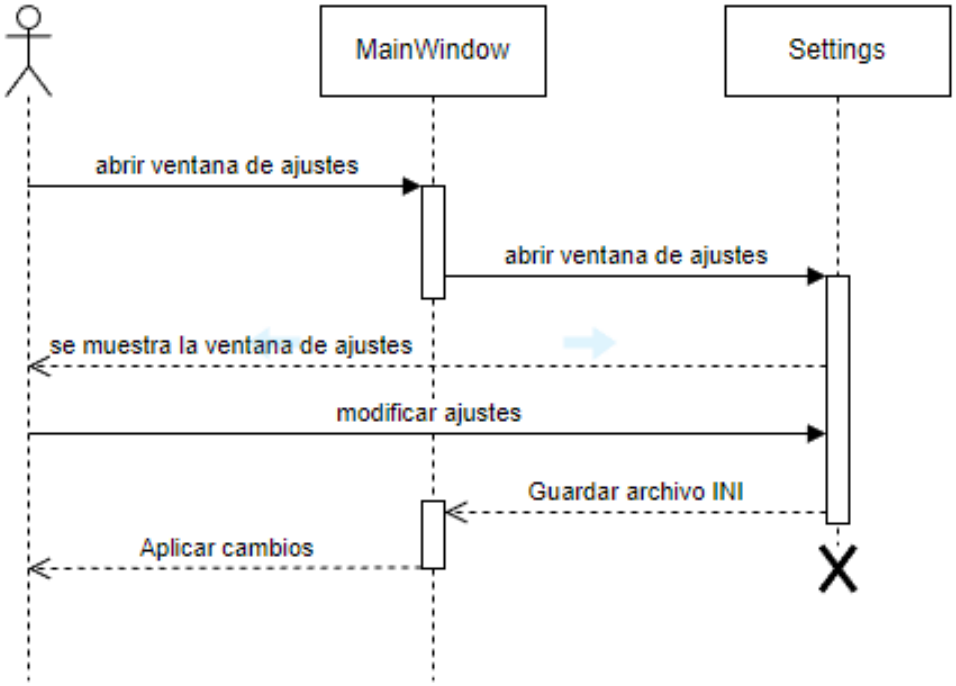

Figura 5.2: Diagrama de secuencia de CU-01: Modificar ajustes

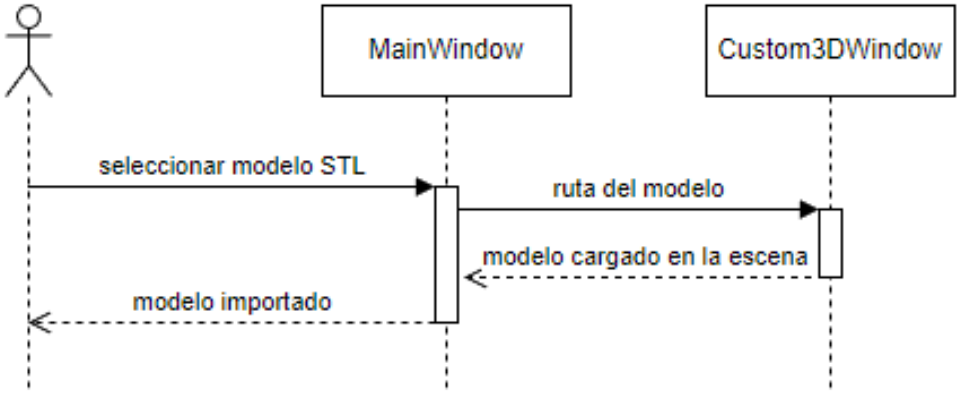

Figura 5.3: Diagrama de secuencia de CU-02: Importar modelo 3D

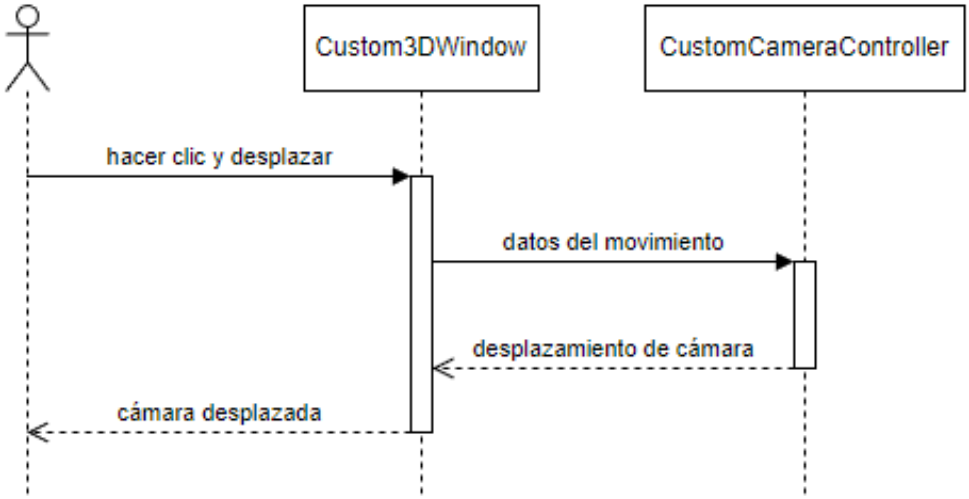

Figura 5.4: Diagrama de secuencia de CU-03: Mover cámara

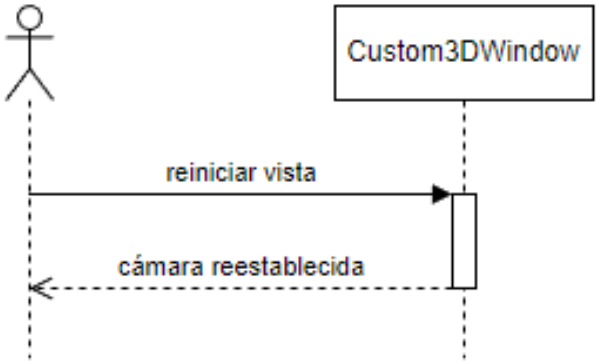

Figura 5.5: Diagrama de secuencia de CU-04: Reiniciar vista

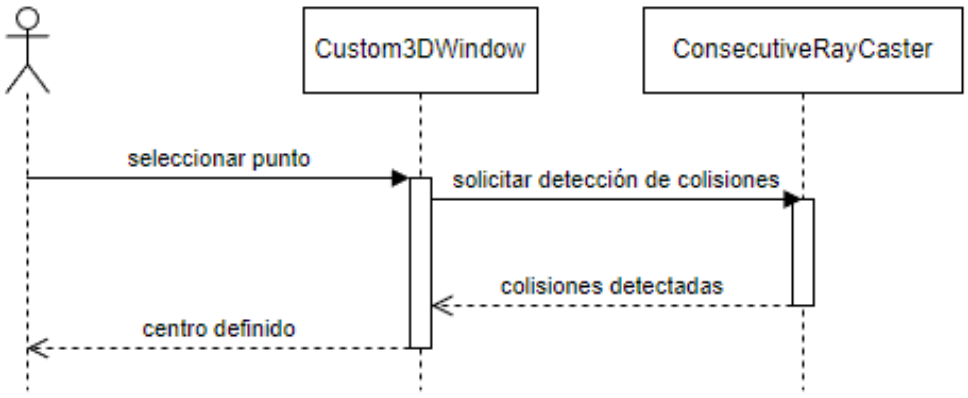

Figura 5.6: Diagrama de secuencia de CU-05: Definir centro

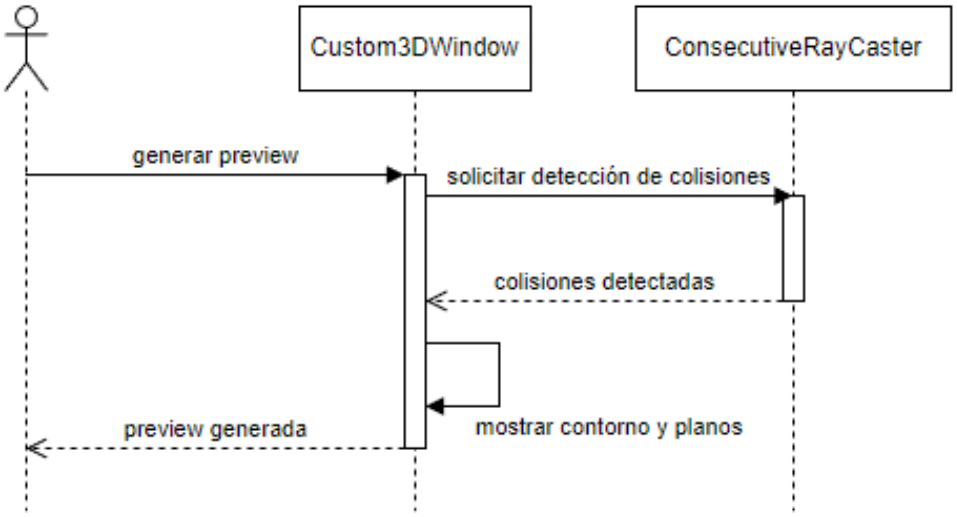

Figura 5.7: Diagrama de secuencia de CU-06: Generar preview

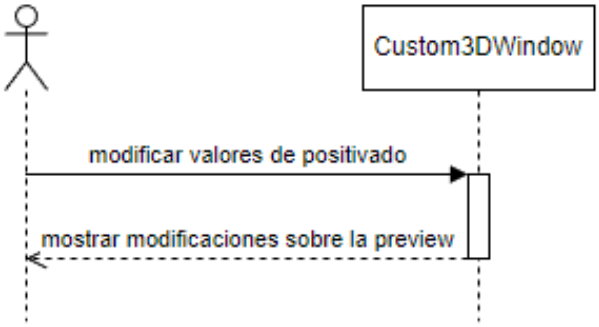

Figura 5.8: Diagrama de secuencia de CU-07: Ajustar positivado

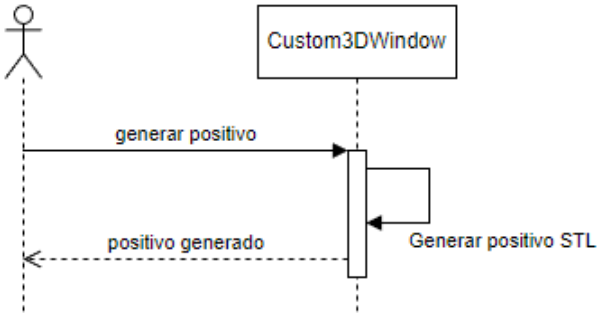

Figura 5.9: Diagrama de secuencia de CU-08: Generar positivo

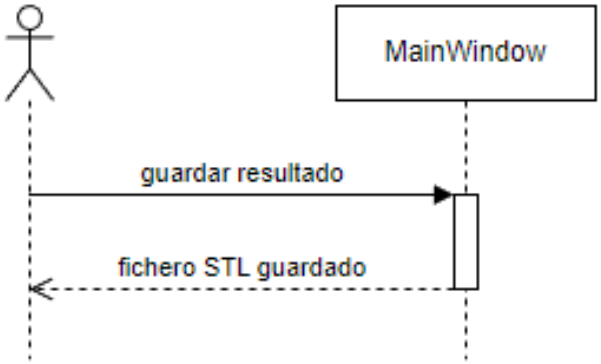

Figura 5.10: Diagrama de secuencia de CU-09: Guardar resultado

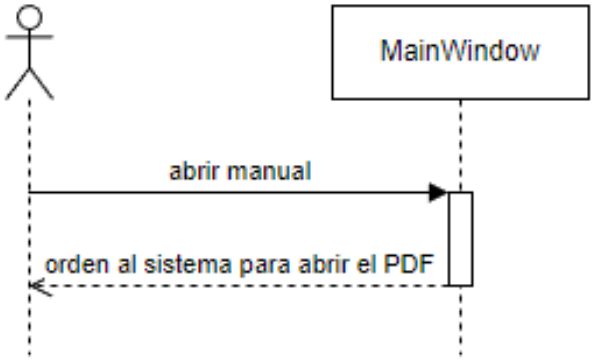

Figura 5.11: Diagrama de secuencia de CU-10: Abrir manuales

5.3. Interfaz gráfica

Durante el diseño de la GUI se establece un sistema de tabs como de secuencia de pasos para definir un flujo de trabajo claro e intuitivo, ya que el perfil de usuario para el que se destina el software cuenta con poca experiencia en el manejo de aplicaciones. La movilidad entre tabs está desactivada por defecto, y sólo es posible cambiar entre ellos utilizando los botones de siguiente o volver en cada tab; esto permite un uso guiado con el objetivo de facilitar la comprensión al usuario y evitar errores. Además, si el resultado obtenido en una fase del flujo no es el esperado, el usuario siempre puede volver al paso anterior y realizar las modificaciones que considere oportunas.

Como se puede ver en la figura 5.12, al iniciar la aplicación no se carga nada y se muestran en la barra de herramientas las dos opciones que más uso tendrán, abrir un archivo y reiniciar la vista. Al importar un modelo el título de la ventana cambiará, indicando la ruta cargada, para informar al usuario durante todo el flujo con qué archivo está trabajando.

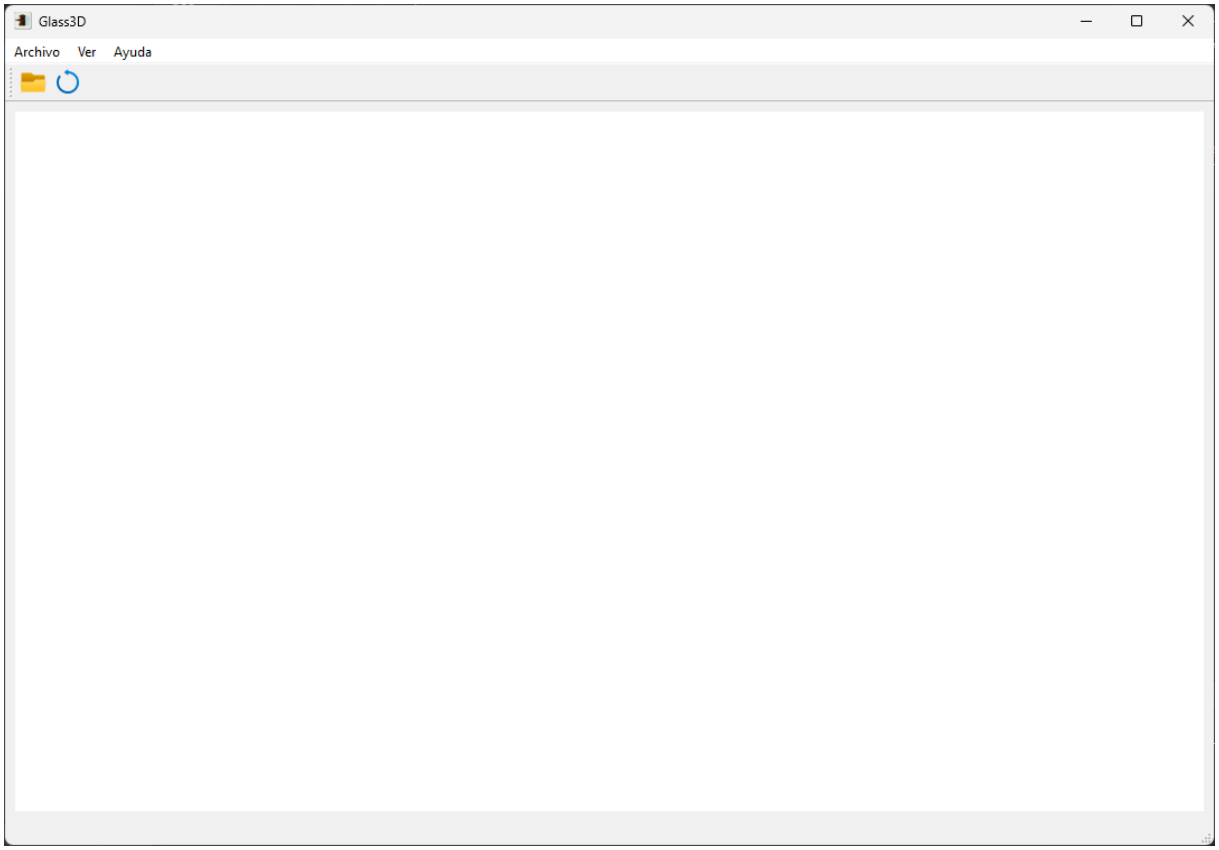

Figura 5.12: GUI: Aplicación recién iniciada

Una vez se ha importado un modelo 3D seleccionando un fichero STL se muestran los

tabs del proceso guiado situándose en el paso 1, para definir un centro. En este paso se puede observar unas pequeñas instrucciones en la parte inferior izquierda, y un campo que indica al usuario la posición del centro definido en la parte inferior derecha. Sólo si está definido el centro se permite al usuario avanzar al siguiente paso; de no ser así se muestra un error indicando que la acción está pendiente. En la figura 5.13 se puede ver una captura del paso 1 antes de avanzar al siguiente.

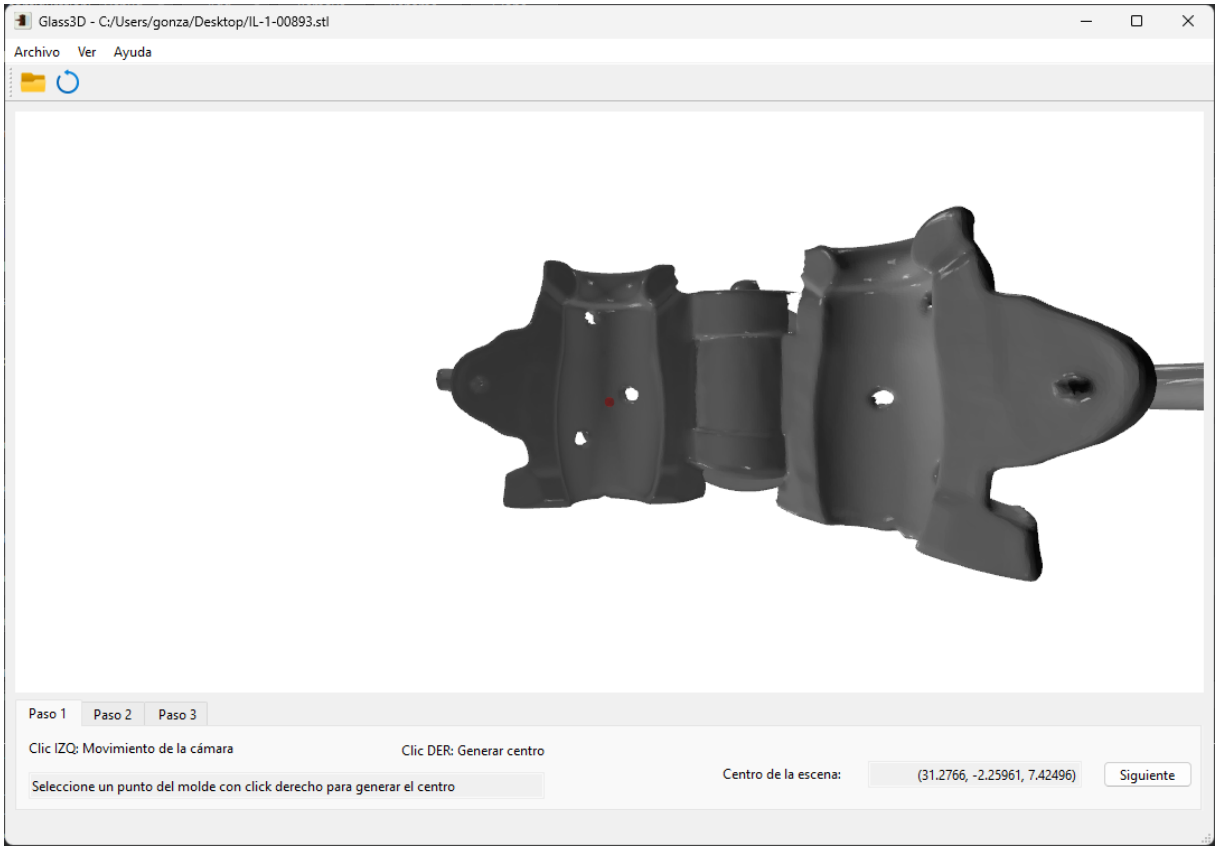

Figura 5.13: GUI: Paso 1 completado

En el paso 2 la primera acción a realizar es pulsar el botón de generar preview, permitiendo al usuario seleccionar en que dirección del eje X quiere realizar el escaneo. En este tab se incluye una barra de progreso en la parte inferior izquierda para informar al usuario del estado de completitud de las funciones en ejecución, ya que el escaneo del contorno y la generación del positivo no serán acciones inmediatas. Durante la ejecución de estas dos funciones más complejas se pausa la interacción del usuario desactivando opciones para evitar errores, una vez han terminado se vuelve a activar todo. En la figura 5.14 se observa el proceso de generación del positivo.

Aún en el paso 2, con la preview cargada se habilita al usuario el desplazamiento de los planos verde y azul; estos pueden moverse modificando su variable de altura manualmente o usando la rueda del ratón sobre esos campos. Una vez se han ajustado los valores de

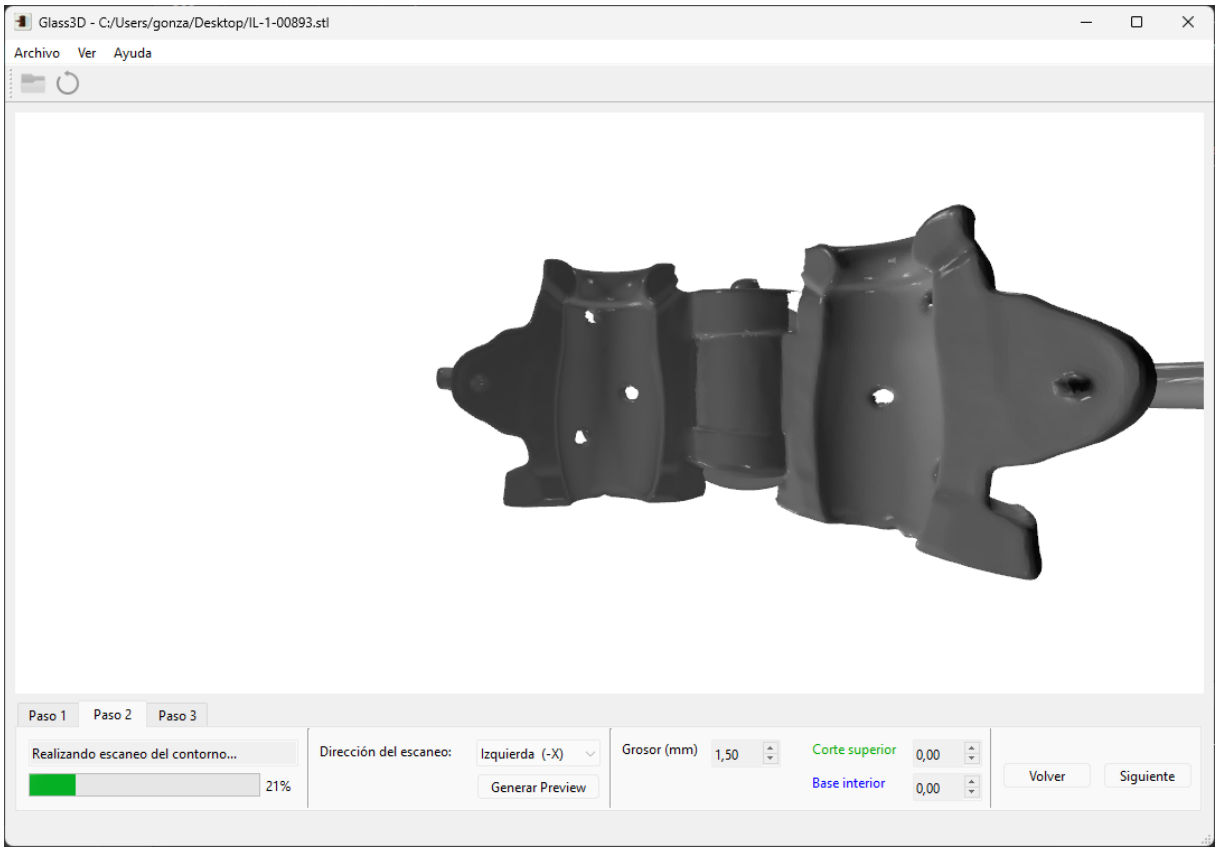

Figura 5.14: GUI: Paso 2 generando preview

generación, se pulsa el botón siguiente para avanzar al paso 3, aunque antes de avanzar debe finalizar el proceso de generación que muestra la barra de progreso de la figura 5.15.

Al avanzar al último tab ya sólo queda comprobar el resultado obtenido en la generación del positivo; para ello se dispone de unos checkbox que muestran u ocultan el molde y/o el resultado, facilitando comprobar la fidelidad del resultado (Figura 5.16). Si el usuario está conforme con el resultado puede guardar la pieza pulsando el botón correspondiente o, en caso contrario, volver a pasos anteriores y modificar parámetros.

La propia interfaz gráfica incluye un acceso rápido a los manuales de la aplicación como se puede ver en la figura 5.17,

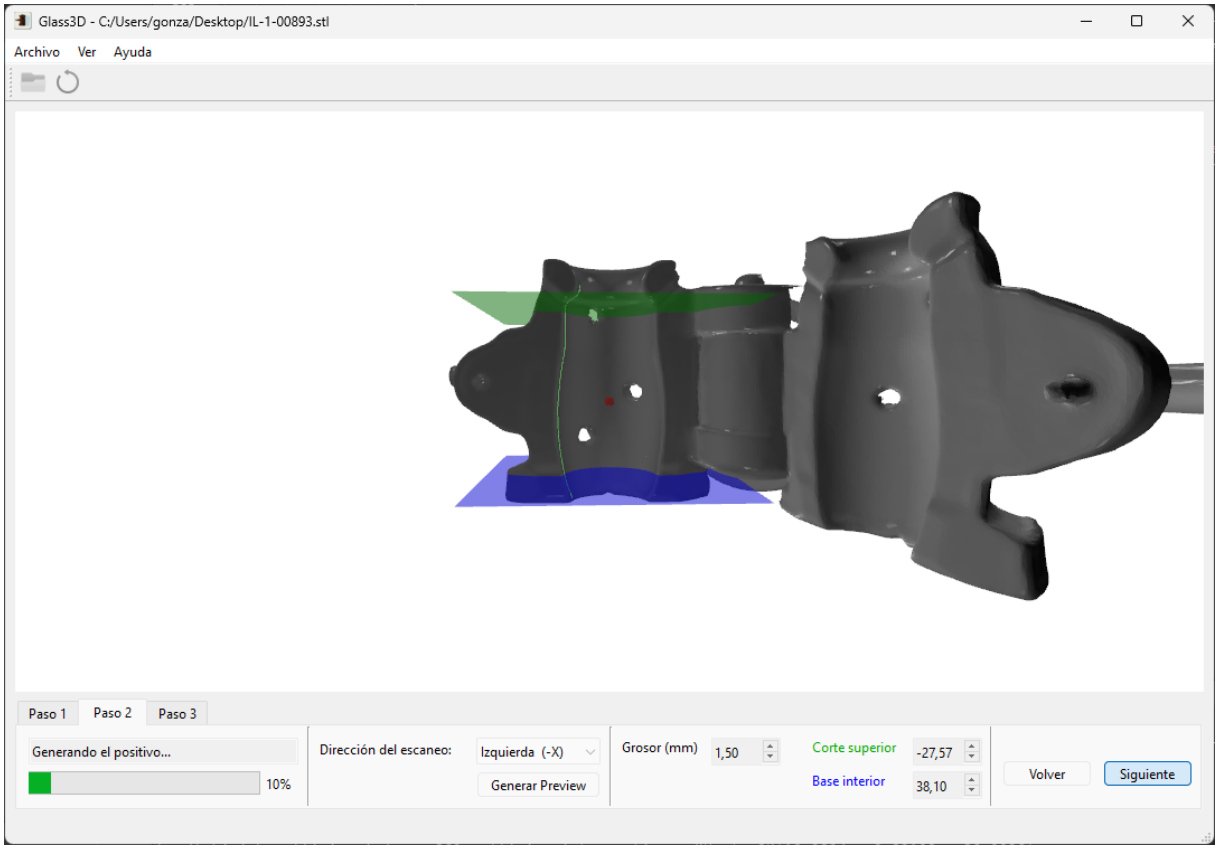

Figura 5.15: GUI: Paso 2 generando positivo

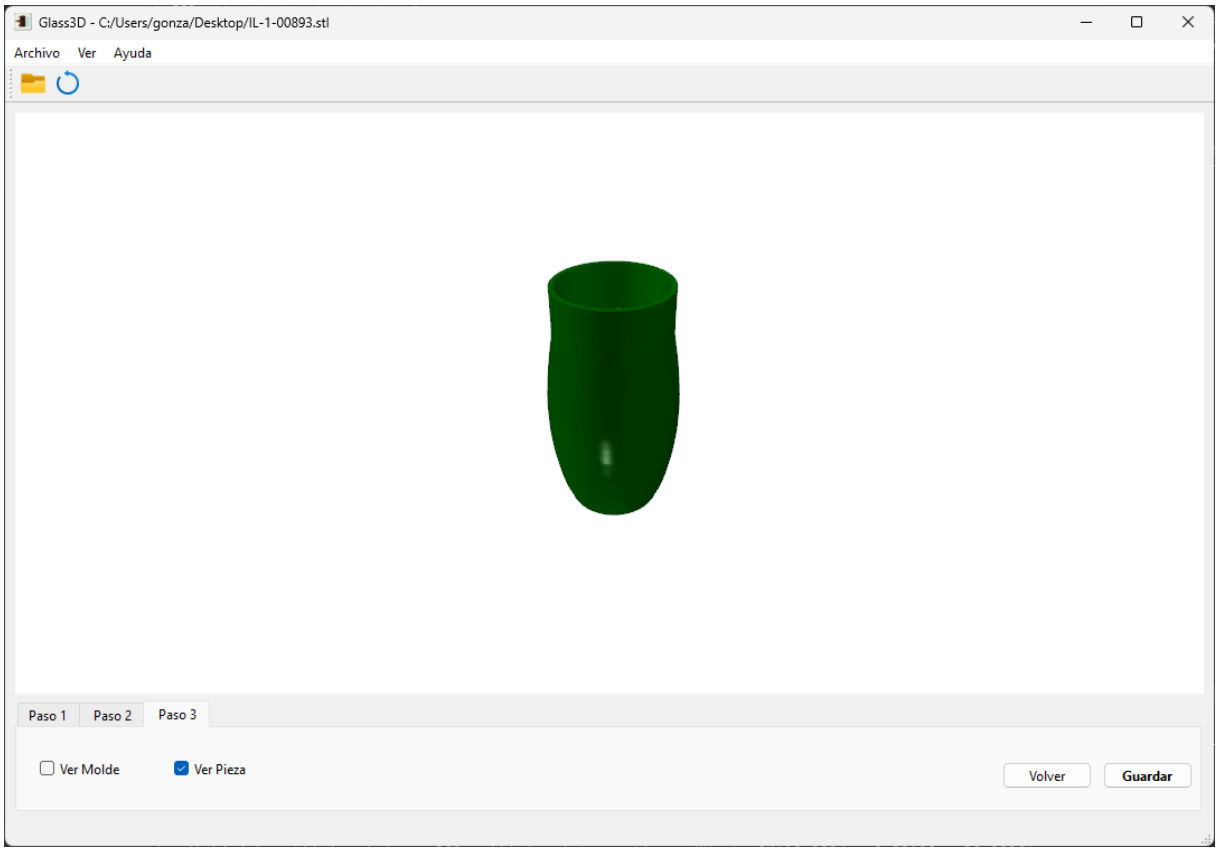

Figura 5.16: GUI: Paso 3 comprobando resultado

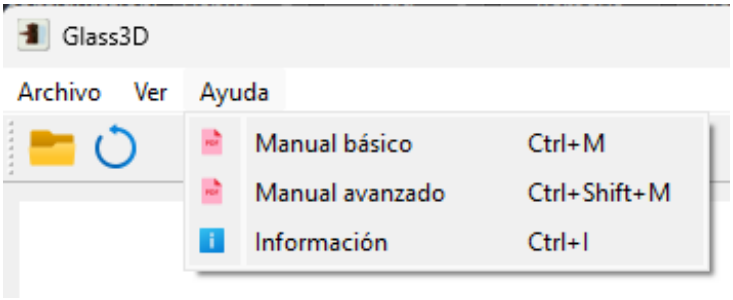

Figura 5.17: GUI: Menú de ayuda

Capítulo 6

Implementación

En este capítulo se mostrará la implementación de las funciones más complejas del desarrollo y sus modificaciones respecto a lo planteado inicialmente.

6.1. Definición del centro

Inicialmente la selección del centro sobre el modelo debía proporcionar únicamente las profundidad a la que se iba a realizar la obtención del contorno del molde, pues se definiría el centro de cada molde en su escaneo. Pero se comprueba que no es posible definir el centro sin seleccionar una parte sólida del modelo, por así decirlo, en el aire. Por tanto, se establece el centro de los moldes en el lateral izquierdo de la pieza, como se puede ver en la figura 6.1.

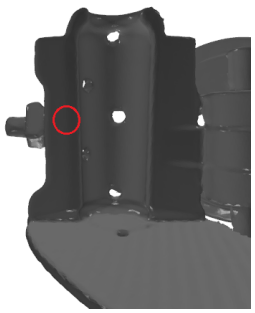

Figura 6.1: Centro inicial de un molde

Para solventar esto se implementa la funcionalidad para calcular el centro a partir de la primera aproximación, que definía la profundidad. Esta mejora realiza dos trazados de rayos en direcciones opuestas del eje X desde el punto que seleccione el usuario con la metodología anterior, es decir, se mantiene el eje Y del punto seleccionado. En la figura 6.2

Figura 6.2: Trazado de rayos para definir el centro

se muestra gráficamente el trazado de estos rayos. En base a las dos colisiones obtenidas obtendremos el propio punto seleccionado en una dirección, y otro punto más alejado en la otra. Calculando la distancia entre estos dos puntos se obtiene el nuevo centro y se reinicia la posición de la cámara en base a éste. En la figura 6.3 se puede ver un ejemplo de selección sobre el molde, que genera el resultado de la figura 6.4.

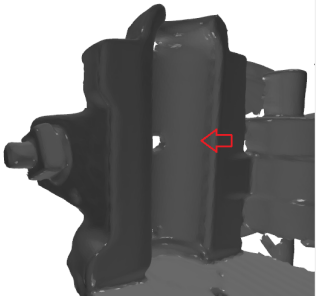

Figura 6.3: Clic sobre la superficie del molde para definir un centro

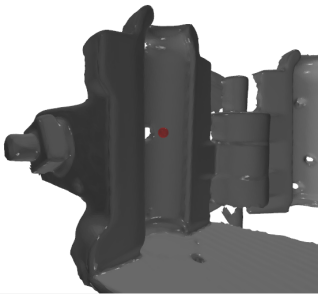

Figura 6.4: Resultado del centro definido por el usuario

Código

En la figura 6.5 se muestra la función *Custom3DWindow::select_points* que se ejecuta al hacer clic sobre el modelo importado; al comprobar que se ha hecho un clic derecho invoca a la función *Custom3DWindow::rayCastCenter* (Figura 6.6). Cuando la función *rayCastCenter* finalice su ejecución llamará a *Custom3DWindow::centerCollisions* para calcular y mostrar en pantalla el centro generado.

```
void Custom3DWindow::select_points(Qt3DRender::QPickEvent *pick)
{
    // Paso 1: Selección del centro del molde
    if (tabSelected == 0 && pick->button() == Qt3DRender::QPickEvent::RightButton) {
        deleteEntities("selectedPoint");
        deleteEntities("centerPoint");
        m_picked = pick->worldIntersection();
        centerDepth = m_picked.y();

        rayCastCenter();
    }
}
```

Figura 6.5: Función de selección de puntos

```
void Custom3DWindow::rayCastCenter()
{
    int originX = static_cast<int>(m_picked.x());
    if (originX < 20)
        originX++;

    m_centerRayCaster = new ConsecutiveRayCaster(rootEntity, TimeDelayStatus::SomeDelay, 0, 2, QVector3D(originX,m_picked.y(),m_picked.z()));
    connect(m_centerRayCaster, SIGNAL(rayCastingFinished()), this, SLOT(centerCollisions()));
    m_centerRayCaster->runRayCastTests();
}
```

Figura 6.6: Función de cálculo del centro

En la figura 6.7 se muestra la función *ConsecutiveRayCaster::runRayCastTests* usada para escanear un número definido de puntos sobre una circunferencia que itera sobre los ejes X y Z. Al momento de escanear el contorno se usa una versión de esta función que escanea puntos, solo en la media circunferencia definida.

```
void ConsecutiveRayCaster::runRayCastTests()
{
    m_rayTests->clear();

    // m_points puntos en el círculo
    int count = m_points;
    for (int i = 0; i < count ; ++i) {
        float tetha = 2 * M_PI / count * i;
        float r = 1000.0f; // Radio
        float x = r * cosf(tetha); // x del punto en el círculo
        float z = r * sinf(tetha); // z del punto en el círculo

        RayTest rayTest = {};
        rayTest.origin = m_center; // Centro
        rayTest.end    = QVector3D( x, m_center.y(), z); // Punto del círculo
        m_rayTests->append(rayTest);
    }

    m_rayCastHandler->runTests(m_rayTests);
}
```

Figura 6.7: Función de escaneo por rayos

6.2. Obtención del contorno

El planteamiento inicial de esta función era seguir los pasos del proceso manual en Blender, crear un plano sobre el centro definido anteriormente y obtener el contorno mediante una operación booleana contra el modelo 3D del molde. En la figura 6.8 se puede observar un ejemplo gráfico de la operación booleana en Qt3D.
Después de investigar sobre el uso de esta posibilidad en Qt y lecturas de documentación se llega a la conclusión de que no es viable. La operación booleana de Qt funciona únicamente en tiempo de renderizado, es decir, no modifica la geometría de ninguno de los modelos usados en el proceso.

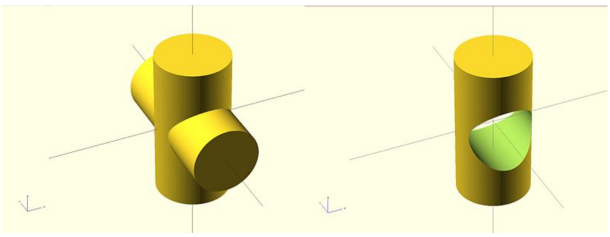

Figura 6.8: Ejemplo de operación booleana en Qt3D

En esta situación se desarrolla el escaneo del contorno mediante el trazado de rayos y detección de colisiones usado en la definición del centro. Esta vez se realizará un escaneo de cientos de puntos en un conjunto de direcciones que forman media circunferencia, empezando por la zona inferior del molde. Tanto la dirección del escaneo como la cantidad de puntos son establecidas por el usuario. En la figura 6.9 se puede ver una representación gráfica de este trazado de rayos hacia la izquierda.
Una vez termina el escaneo se usan las colisiones obtenidas para dibujar el contorno resultante, además de los planos de corte superior y de generación de la base interior. Este resultado se muestra en la figura 6.10.

Código
Al pulsar el botón generar preview se ejecuta el método *Custom3DWindow::loadGeometry*, que inicia el escaneo de puntos en forma de media circunferencia para definir el contorno. Esta función se ejecuta sobre un nuevo hilo para no bloquear el hilo principal de la aplicación y así mostrar el avance en un barra de progreso de la GUI. Esta función se muestra en la figura 6.11.
Al terminar la ejecución el escaneo por rayos se recibe una señal que invoca a la función *Custom3DWindow::previewCollisions*, que se encarga de dibujar el contorno uniendo líneas entre los puntos detectados en el escaneo y genera los planos.

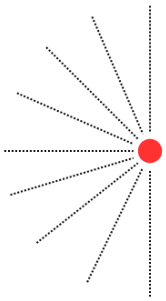

Figura 6.9: Trazado de rayos para escanear el contorno

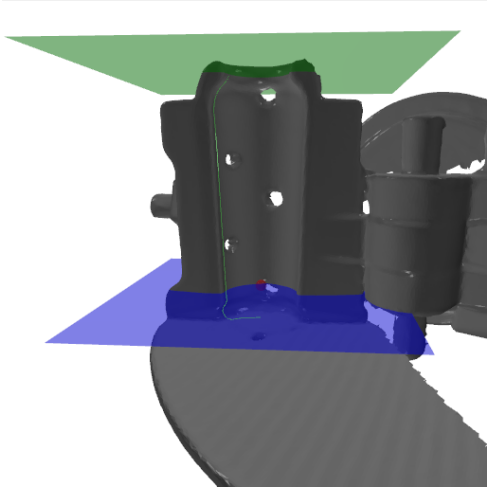

Figura 6.10: Resultado de la obtención del contorno

```
void Custom3DWindow::loadGeometry()
{
    m_centerSphere->setEnabled(false);
    emit setInfoStep2("Realizando escaneo del contorno...");

    m_positiveRayCaster = new ConsecutiveRayCaster(rootEntity, TimeDelayStatus::NoDelay, 0 /* milliseconds */, rayCastCount*2, m_center);
    m_positiveRayCaster->setHalfDirection(rayCastDirection);
    connect(m_positiveRayCaster, SIGNAL(rayCasterProgressBar(int)), this, SLOT(emitProgressBar(int)));

    // Una vez ha terminado el escaneo de puntos emito la señal de finalización para ejecutar el slot que crea los triángulos
    connect(m_positiveRayCaster, SIGNAL(rayCastingFinished()), this, SLOT(previewCollisions()));

    QThread *thread = new QThread;
    m_positiveRayCaster->moveToThread(thread);
    thread->start();

    m_positiveRayCaster->runHalfRayCastTests();
}
```

Figura 6.11: Función de escaneo de la preview

6.3. Generación del positivo

6.3.1. Formato STL

Antes de abordar la implementación del paso final se debe entender como funciona el formato de archivo STL a la hora de leer los triángulos y la posición de sus vértices, para ello se divide este apartado en los dos formatos de escritura que se pueden usar: ASCII y binario.

ASCII
En este formato se introducen los datos mediante la escritura de texto plano, incluyendo primero una cabecera y luego un bucle "facet" por cada triángulo para definir la posición de los vértices y la normal de este. El uso de texto plano tiene algunas desventajas frente a la escritura binaria, como el tamaño de archivo y los tiempos de procesado por programas de modelado como Blender. La gran ventaja por la que se usa en este escenario es que no requiere introducir el número total de triángulos que componen el modelo, valor desconocido durante el proceso de generación. En la figura 6.12 se puede ver un ejemplo escrito en este formato donde se muestra la cabecera y un triángulo.

```
solid PositivoTemporal
facet normal 0 0 -1
outer loop
vertex 0 0 -78.1112
vertex 0.0813856 0.0264438 -78.1112
vertex 0.0855739 0 -78.1112
endloop
endfacet
```

Figura 6.12: Ejemplo de fichero STL ASCII

Binario
El formato binario es el más usado y ocupa un menor tamaño en disco. Para componer un fichero se deben cumplir las condiciones de longitud en bytes que especifica el formato, mostradas en la figura 6.13.

El dato del que no se dispone para usar únicamente este formato es el número total de triángulos pero, en la siguiente sección se explica como se ha logrado obtener. En la figura 6.14 se muestra un ejemplo de fichero STL en formato binario.

```
UINT8[80]    - Header                 -     80 bytes
UINT32       - Number of triangles    -      4 bytes

foreach triangle                      - 50 bytes:
    REAL32[3] - Normal vector             - 12 bytes
    REAL32[3] - Vertex 1                  - 12 bytes
    REAL32[3] - Vertex 2                  - 12 bytes
    REAL32[3] - Vertex 3                  - 12 bytes
    UINT16    - Attribute byte count      -  2 bytes
end
```

Figura 6.13: Formato STL Binario: longitud de campos

```
Address  0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 73 6f 6c 69 64 20 62 69 6e 54 65 6d 70 2e 73 74 solid binTemp.st
00000010 6c 2d 6f 75 74 70 75 74 00 00 00 00 00 00 00 00 l-output........
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000050 c0 0d 00 00 00 00 00 00 00 00 00 00 00 00 80 bf À..............€¿
00000060 00 00 00 00 00 00 00 00 ef 38 9c c2 7e ad a6 3d ........ï8œÂ~-¦=
00000070 ab a0 d8 3c ef 38 9c c2 5e 41 af 3d 00 00 00 00 «.Ø<ï8œÂ^A¯=....
00000080 ef 38 9c c2 61 61 94 3f fc bc f8 ce 9f bb 23 e0 ï8œÂaa”?ü¼øÎŸ»#à
00000090 7f bf 29 ed 95 40 00 00 00 00 7c 81 9c c2 7e ad .¿)í•@....|.œÂ~-
000000a0 a6 3d ab a0 d8 3c ef 38 9c c2 a7 96 8e 40 d7 51 ¦=«.Ø<ï8œÂ§–Ž@×Q
```

Figura 6.14: Ejemplo de fichero STL Binario

6.3.2. Generación

Todo el proceso de generación se lleva a cabo de forma simultánea. A partir de cada punto del contorno se generan puntos en una circunferencia tomando como radio la distancia entre el punto y el centro definido, teniendo en cuenta únicamente los valores del eje X. El número de puntos que se generan en cada circunferencia es definido por el usuario.
Durante la generación de estas circunferencias se hacen uniones de dos triángulos cada cuatro puntos, a excepción de las bases del positivo que se genera un triángulo cada tres puntos. En las figuras 6.15 y 6.16 se pueden ver la interfaz previa a la generación y el resultado, respectivamente. También se puede observar en la figura 6.17 como se componen los triángulos de las paredes.

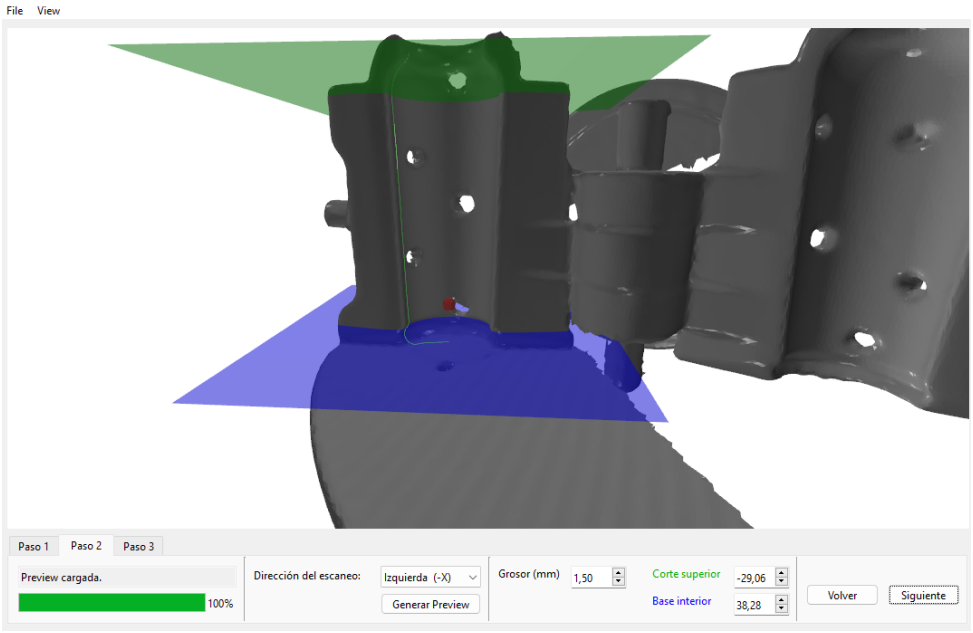

Figura 6.15: Interfaz previa a la generación de un positivo

Figura 6.16: Resultado de un positivo generado

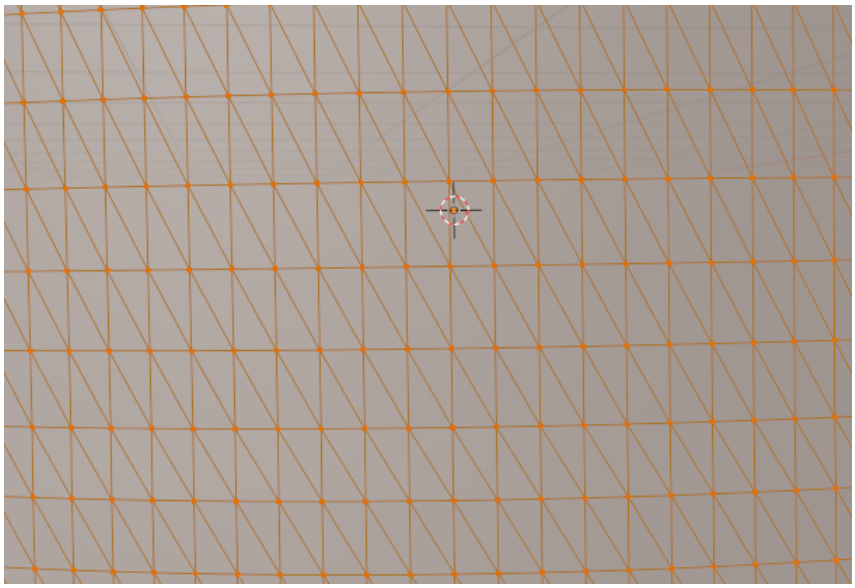

Figura 6.17: Composición triangular de una pared

Teniendo en cuenta que la generación del positivo comienza por la zona inferior del molde, las variaciones definidas por el usuario como el grosor y los planos se gestionan de la siguiente forma:

- **Grosor:** Este valor decimal establece en milímetros la distancia entre el radio que se usa para componer la pared exterior y el radio de la pared interior.
- **Plano de la base interior:** Este plano está definido de color azul en la GUI y se usa para establecer un valor de altura desde el que empezar a componer el interior de la pieza. La primera colisión que supere esta altura generará la base interior y a partir de ella se generará la pared interior al mismo tiempo que la exterior, distanciándose en la medida de grosor definida anteriormente. En la figura 6.18 se puede observar la generación de esta base uniendo cada dos puntos de la circunferencia con el centro, sin modificar la altura del punto del contorno.
- **Plano de corte superior:** Este plano, de color verde en la GUI, establece la altura máxima que no pueden superar los puntos del contorno, es decir, los puntos que se encuentren por encima serán ignorados en el proceso de generación. Además, en el último punto posible antes de superar este plano se compone el cierre de las dos paredes, como se puede observar en la figura 6.19.

Conviene destacar que, aunque en el ejemplo se muestra un molde con base, si se tratase de uno que no la posea, la generación de la base exterior se haría exactamente igual que para la base interior. Se compondría una base plana desde el centro hasta el primer punto del contorno. En la figura 6.20 se puede ver la composición de un molde con base.

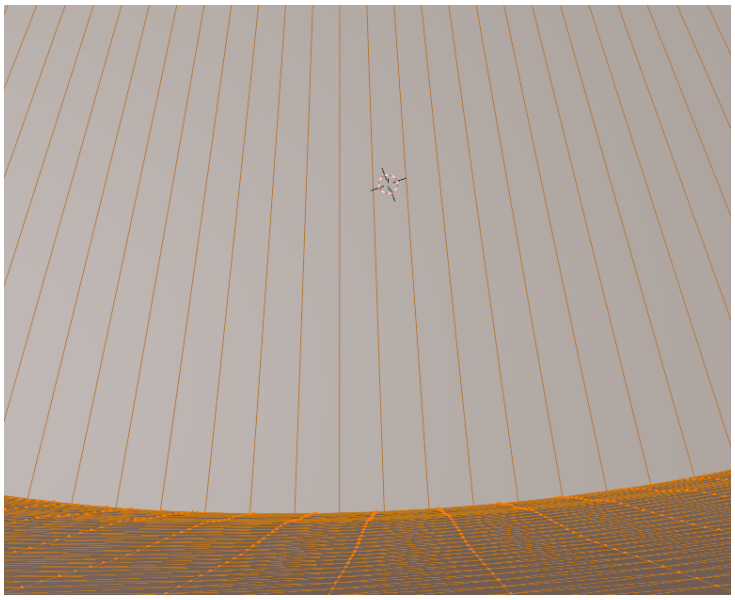

Figura 6.18: Composición triangular de la base interior

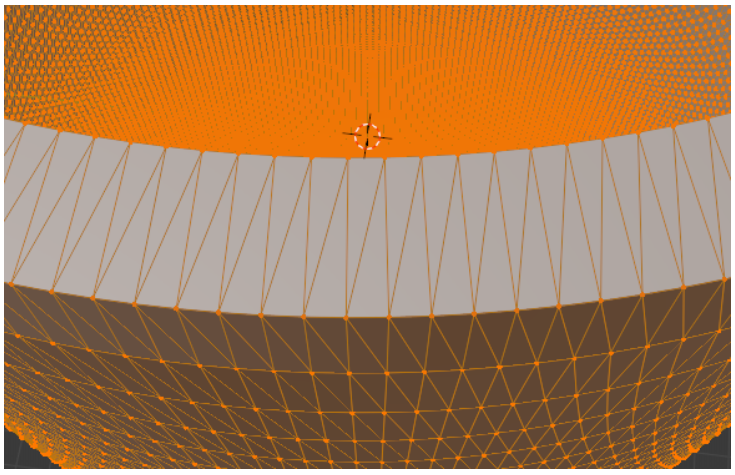

Figura 6.19: Composición triangular del corte superior

Código

La función encargada la generación del positivo es *Custom3DWindow::positiveCollisions*, una función muy extensa que itera sobre los puntos escaneados realizando sobre cada uno su revolución circular y uniendo triangulos con la revolución formada por el siguiente punto. En la figura 6.21 se puede ver la parte final del bucle, donde se cierra la parte superior uniendo la pared interna con la externa. Además, también se puede ver en esta sección del código cómo se van escribiendo los triángulos en cada iteración del bucle para no saturar la escritura al final, ahorrando así memoria de procesado y evitando posibles errores por intentar una carga masiva de triángulos.

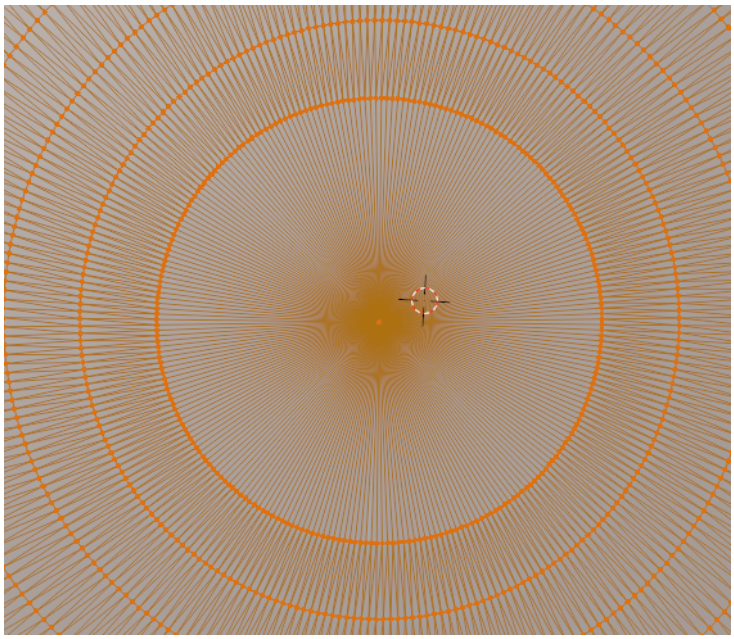

Figura 6.20: Composición triangular de la base exterior

Una vez ha finalizado la generación se ha creado un fichero temporal en formato STL ASCII, que Glass3D importa para mostrar el resultado obtenido en la escena. Aprovechando las funciones que nos ofrece el objeto *Qt3DRender::QMesh* para estudiar la geometría de un modelo importado obtenemos el número total de triángulos, pudiendo exportar finalmente el resultado en formato STL binario. La función encargada de este último proceso en el momento de guardar el resultado es *Custom3DWindow::exportBinarySTL(QMesh)*.

```
    } else if (!positivoCerrado) { // CIERRE DE LA PARTE SUPERIOR (usando plano de corte superior)
        // Utilizo los valores guardados de los puntos "Next" para realizar el cierre.
        for (int j = 0; j < trianglesCount ; j++) {
            // Obtengo los valores para el sin y cos del punto j del circulo y el siguiente
            tetha = 2 * M_PI / trianglesCount * j;
            if (j==trianglesCount-1) {
                tethaAux = 2 * M_PI / trianglesCount * 0;
            } else {
                tethaAux = 2 * M_PI / trianglesCount * (j+1);
            }

            // EXTERIOR
            xNext = rNext * cos(tetha);
            yNext = rNext * sin(tetha);
                // Siguiente punto en el círculo
            xNextAux = rNext * cos(tethaAux);
            yNextAux = rNext * sin(tethaAux);

            // INTERIOR
            xIntNext = rIntNext * cos(tetha);
            yIntNext = rIntNext * sin(tetha);
                // Siguiente punto en el círculo
            xIntNextAux = rIntNext * cos(tethaAux);
            yIntNextAux = rIntNext * sin(tethaAux);

            meshExport.add(Vec3(xNext,yNext,-intersection.z()),Vec3(xNextAux,yNextAux,-intersection.z()),Vec3(xIntNextAux,yIntNextAux,-intersection.z()));
            meshExport.add(Vec3(xIntNextAux,yIntNextAux,-intersection.z()),Vec3(xIntNext,yIntNext,-intersection.z()),Vec3(xNext,yNext,-intersection.z()));
        }
        positivoCerrado = true;
    }
    meshExport.stl_write_triangles("temp.stl");
    meshExport.clear();
}
meshExport.stl_write_end("temp.stl","PositivoTemporal");
```

Figura 6.21: Función de generación del positivo (parte final)

Capítulo 7

Pruebas

En este capítulo se recopilan las pruebas y validaciones desarrolladas sobre la aplicación para asegurar su buen funcionamiento y evitar errores.

7.1. Pruebas de caja negra

El uso de Qt como framework encargado de la interfaz gráfica ha facilitado considerablemente la tarea de validación de los campos modificables por el usuario, pues Qt permite establecer valores máximos y mínimos, que en ningún momento pueden superar esos campos aunque se estableciese por código.
Validaciones de campos en el flujo de trabajo:

- **Dirección del escaneo:** No se permite la edición de esta lista desplegable ni la selección de un valor no definido, izquierda o derecha.
- **Grosor de la pieza generada:** el valor mínimo de este campo es 0.05 milímetros, mientras que el valor máximo no está definido para permitir el escaneo de moldes de cualquier tamaño.
- **Planos:** Los dos planos que puede modificar el usuario no están expuestos a limitaciones, pero se ha comprobado que los valores extremos no dan error en la generación del positivo.

Aunque estos valores no tengan limitaciones y no generen errores, la modificación sin sentido de estos campos provocará una generación con poca fidelidad al molde escaneado.
Validaciones de campos en las preferencias:

- **Dirección de escaneo y grosor:** Comparten las reglas de validación descritas anteriormente.
- Puntos escaneados en preview: Se establece un valor mínimo de 50 y un máximo de 2000.

- **Puntos de generación circular:** Se establece un valor mínimo de 20 y un máximo de 500.

El movimiento entre las fases del flujo de trabajo está limitado para usuario y no se permite avanzar sin haber realizado las acciones necesarias en la fase actual, aunque sí se permite retroceder. En caso de que el usuario intente avanzar sin cumplir los requisitos se le muestra un mensaje por pantalla indicando lo que le falta por hacer.

7.2. Pruebas unitarias

Durante estas pruebas se han comprobado los valores máximos del escaneado y la preview, además de comprobar las optimizaciones aplicadas sobre la función que define el centro.

- **Definición del centro:** Se ha comprobado que las optimizaciones aplicadas a la función reducen considerablemente que se genere el centro en una zona no deseada. Además, en caso de que el usuario seleccione un punto desde el que no se pueden detectar colisiones a los lados, se muestra por pantalla un mensaje informando de la situación.
- **Generación de máxima precisión:** Se ha comprobado que la aplicación es capaz de usar los valores máximos configurables por el usuario, de 2000 puntos de escaneo y 500 de generación circular. La ejecución de estas pruebas resulta en un tiempo de ejecución máximo de 2 minutos 50 segundos para un equipo de gama media y 5 minutos para uno de gama baja.

Capítulo 8

Conclusiones

En este capítulo se habla de la consecución de objetivos y el aprendizaje y satisfacción obtenidos durante su desarrollo.

El aprendizaje y uso de C++ ha sido bastante intuitivo gracias a la formación recibida en el lenguaje C y el paradigma de programación orientada a objetos a lo largo del grado. Después del desarrollo de este software y habiendo trabajado con Qt, se puede concluir que estamos ante un framework con muchas posibilidades para el diseño de interfaces. El desarrollo de la GUI y sus modificaciones ha sido muy intuitivo gracias a las ventanas de diseño que tiene el IDE Qt Creator, y también tiene una gran potencia ejecutando funciones complejas de muchas iteraciones en tiempos reducidos. Aunque Qt3D está pensado para la visualización e interacción de modelos 3D, es esta potencia lo que ha permitido desarrollar funciones de modelado a bajo nivel iterando sobre los miles de triángulos que componen un modelo.

La generación de positivos definiendo los triángulos sobre el propio archivo a exportar resultaba una idea muy compleja inicialmente, pero ha facilitado enormemente la conexión con los parámetros que se le permiten modificar al usuario en la generación. La importancia de incluir esta posibilidad de modificar parámetros es la mayor ventaja de la aplicación, pues permite exportar positivos distintos desde un mismo molde, al igual que se haría durante la confección artesanal del vidrio, dependiendo de quien lo esté trabajando.

Alguna de las desventajas de la generación es la base interior del molde, pues se genera una superficie plana cuando en la fabricación pueden hacerse con alguna forma distinta. Otra limitación puede ser que los moldes tengan mucho detalle en todas las altura; al realizarse el escaneo desde un solo punto central las zonas escaneadas con más precisión serán las próximas a este, pudiendo dejar alguna parte lejana menos definida. Todos estos aspectos contemplan su solución en la sección de líneas de trabajo futuras.

8.1. Conclusión personal

Al inicio del desarrollo tomaba como punto determinante del éxito elegir una buena librería para manejar las entidades 3D dentro de Qt. Esta tarea de investigación fue sin duda la más tediosa, teniendo que invertir muchas horas en un proceso al que no le veía un resultado próximo. Después de consultar multitud de librerías de terceros tomo la decisión de usar Qt3D y, con su uso, me voy dando cuenta de que fue una buena elección. Durante el proceso de desarrollo iba descubriendo funciones no estudiadas en la iteración inicial pero de gran utilidad para mi proyecto. Qt3D fue elegido en parte por estar incluido de forma nativa en Qt, pero también por la gran colección de clases de las que dispone. Mientras más objetivos se cumplían con esta librería más me daba cuenta de que las tediosas horas de investigación no habían caído en saco roto.

Qt es un framework con mucha potencia que me ha permitido realizar pruebas de mucha carga, ayudándome así a entender mejor su funcionamiento en momentos iniciales del proyecto. Aunque me he encontrado baches en el camino, como la imposibilidad de hacer una operación booleana modificando la estructura de un modelo, la situación forzada de cambiar la forma de obtención del contorno ha dado un resultado con más opciones de modificación para el usuario.

Estoy muy contento con el resultado final, pues viene de un duro proceso de investigación del que partía con muchas incógnitas y poca información que consultar, habiendo tenido incluso que consultar el código fuente del propio Qt3D para comprender cómo hacía algunas cosas. En lo personal me ha resultado un proceso enriquecedor porque partía prácticamente a ciegas y me he demostrado a mí mismo que con una buena rutina de trabajo, aunque algunas jornadas fuesen desesperantes, he podido sacar el software que me propuse en un principio. Aunque hay cosas que se han quedado en el tintero y me gustaría haber podido terminar, sé con seguridad que con más tiempo voy a poder terminarlas, pues dispongo de la experiencia y los conocimientos necesarios para realizar un desarrollo fluido sobre las ideas que me vayan surgiendo.

Por último, con el propósito de salir al mundo laboral habiendo terminado con éxito mis estudios de grado, después de haber realizado este proyecto me planteo buscar un primer puesto de trabajo desarrollando sobre C++.

8.2. Líneas de trabajo futuras

En esta sección se plantean las líneas de trabajo planificadas en el momento de finalización de este TFG.

- **Escaneo desde múltiples puntos:** Con el objetivo de no perder detalle en moldes muy complejos se permite al usuario crear centros auxiliares desde los que escanear el contorno, repartiendo la carga total de puntos entre el número de centros definidos.
- **Formas de la base interior:** Incluir una lista desplegable en la interfaz gráfica desde la que poder elegir un tipo de forma para la base interior de un positivo, definiendo también la profundidad.
- **Grosor adaptativo:** Permitir al usuario definir un grosor del borde generado que se encoja o ensanche en función de dos valores de inicio y fin que se indiquen.
- **Cálculo de volumen:** Calcular el volumen de la pieza generada y guardarla en la cabecera del archivo STL exportado. Este dato resulta útil para la FCNV a la hora de catalogar piezas.
- **Escaneo de moldes de soplado fijo:** Permitir el escaneo completo de una cara del molde utilizando el trazado de rayos en vez de ser usado este para obtener sólo el contorno. Una vez obtenida esta mitad de la pieza se podría espejar para obtener el resultado final, aumentando las posibilidades de uso de la aplicación.

Parte III

Manuales de la Aplicación

Capítulo 9

Manual de Usuario

En este capítulo se muestran los manuales a los que puede acceder el usuario para aprender a usar la aplicación. Ambos manuales son accesibles desde el menú de ayuda. La instalación de la aplicación no requiere el uso de ningún software auxiliar, únicamente debe copiarse la carpeta a la ruta del equipo deseada y ejecutarla.

9.1. Manual Básico

En este manual se muestran los pasos para hacer un uso rápido de la aplicación, explicando el flujo de trabajo de ésta. El movimiento de un paso a otro dentro de la aplicación es bidireccional, pudiendo volver a un paso anterior a modificar parte del flujo.

9.1.1. Iniciar el proceso

Para iniciar el flujo de la aplicación lo primero es importar un molde 3D en formato STL. Abrir un fichero es igual a reiniciar el proceso, por ello se puede acceder a esta función desde cualquier parte del proceso. Para ello se debe:

1. Seleccionar la opción del menú superior, **Archivo** → **Abrir**. Figura 9.1.
2. Seleccionar el archivo en cuestión en la ventana emergente.
3. Aceptar

Al terminar, si el fichero no está corrupto, se cargará en la escena principal el molde seleccionado. Para modificar la posición y/o orientación de la cámara debe hacer clic izquierdo sobre la escena y desplazar el cursor simultáneamente.

9.1.2. Paso 1: Definir el centro

En este paso se define el centro desde el que se escanea el contorno del molde en el paso 2. Los pasos para hacerlo son los siguientes:

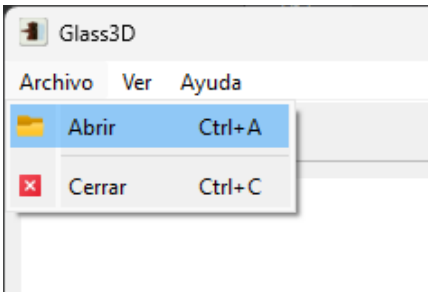

Figura 9.1: Opción de menú para abrir un modelo

1. Hacer clic derecho sobre el modelo importado. Figura 9.2

2. Comprobar la posición del centro generado. Figura 9.3

3. (Opcional) Modificar la posición del centro repitiendo el primer paso.

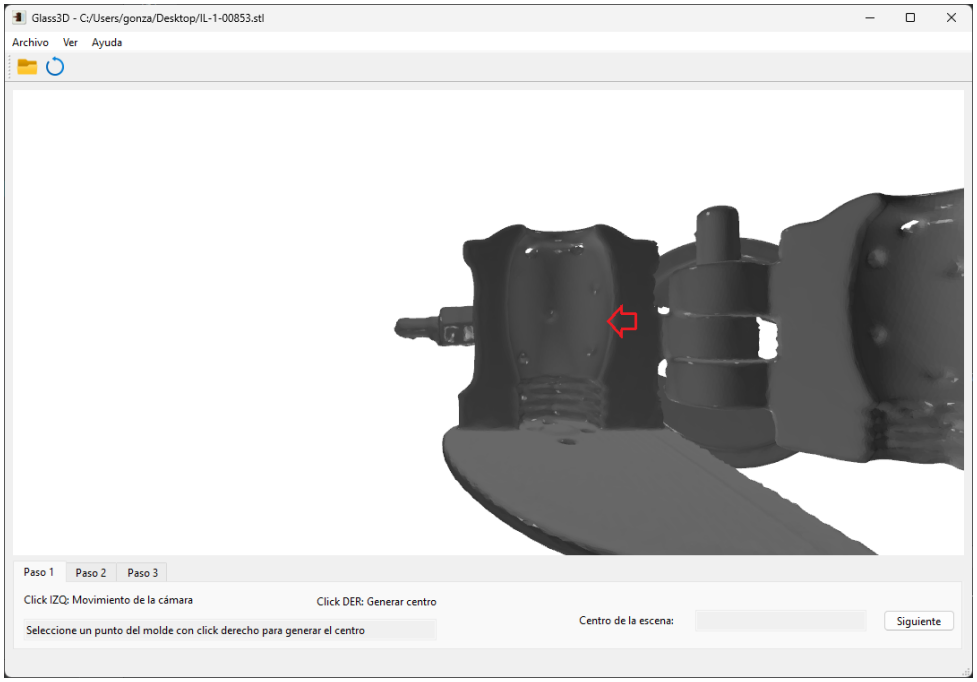

Figura 9.2: Definir centro: Clic derecho sobre el modelo

Si el centro no se puede generar desde el punto seleccionado se muestra un mensaje de aviso que así lo indica. Una vez existe un centro definido se debe pulsar el botón "Siguiente" para avanzar al paso 2.

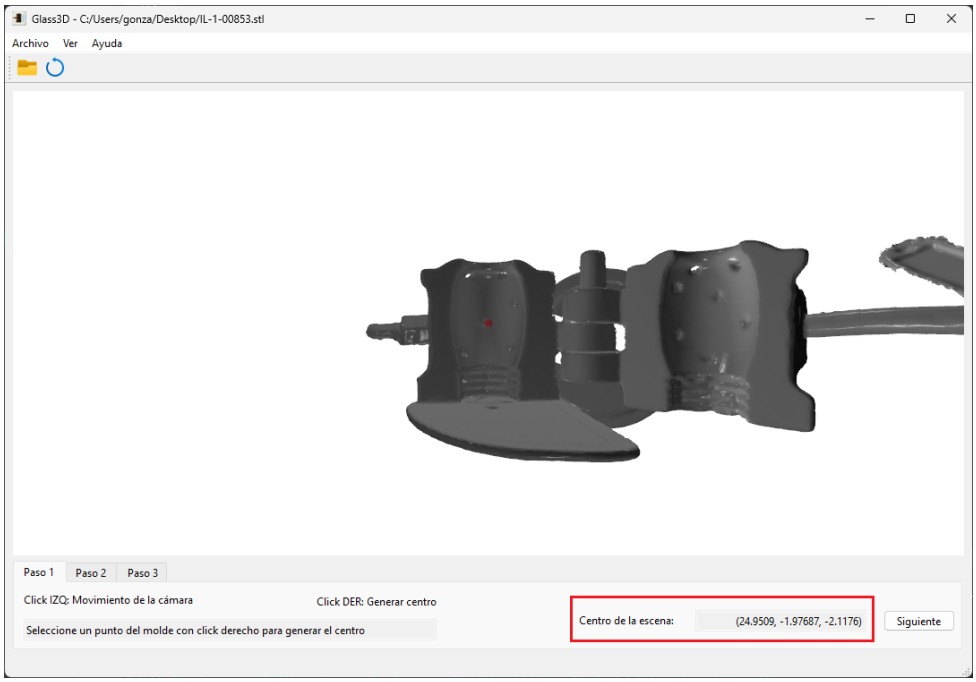

Figura 9.3: Definir centro: Comprobar centro generado

9.1.3. Paso 2: Generar Preview

En este paso se generar una preview que sirve al usuario para comprobar el contorno que se escanea desde el centro que definió en el paso anterior. El escaneo puede definirse en dos direcciones, izquierda o derecha. Para realizar este proceso se debe:

1. Seleccionar la dirección de escaneo.

2. Pulsar el botón de "Generar Preview". Figura 9.4.

9.1.4. Paso 3: Generar Positivo

Una vez se ha generado la preview se pueden realizar ajustes para la generación del positivo en esa misma sección de la aplicación, como se puede ver en la figura **??**.

El proceso de generación sería el siguiente:

1. Modificar los ajustes de generación que se estimen oportunos.

2. Pulsar el botón "Siguiente" para generar el positivo.

3. Comprobar la pieza resultante.

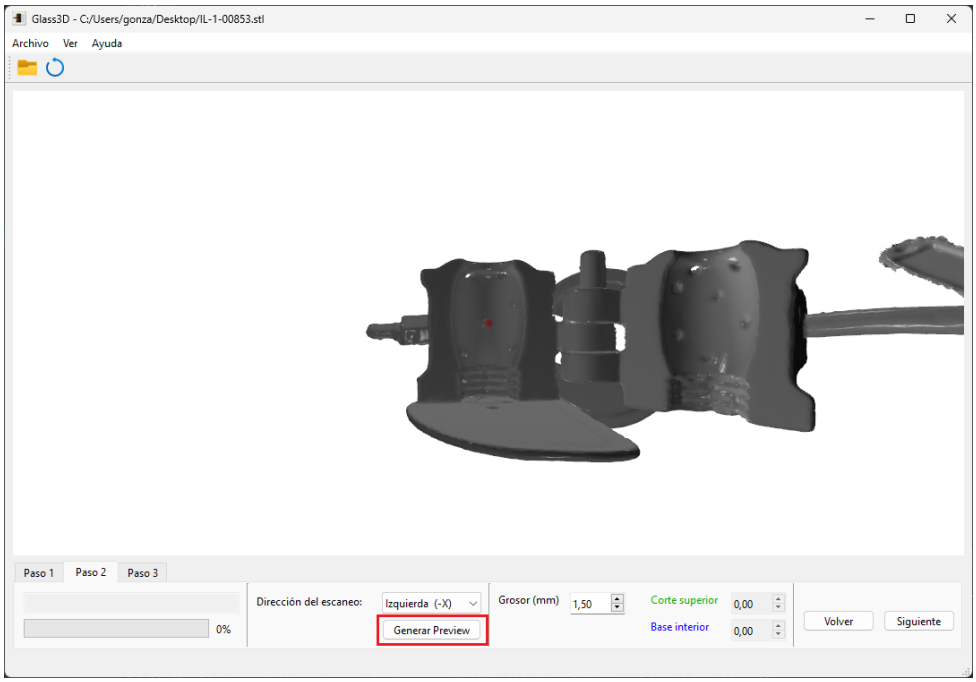

Figura 9.4: Generar Preview

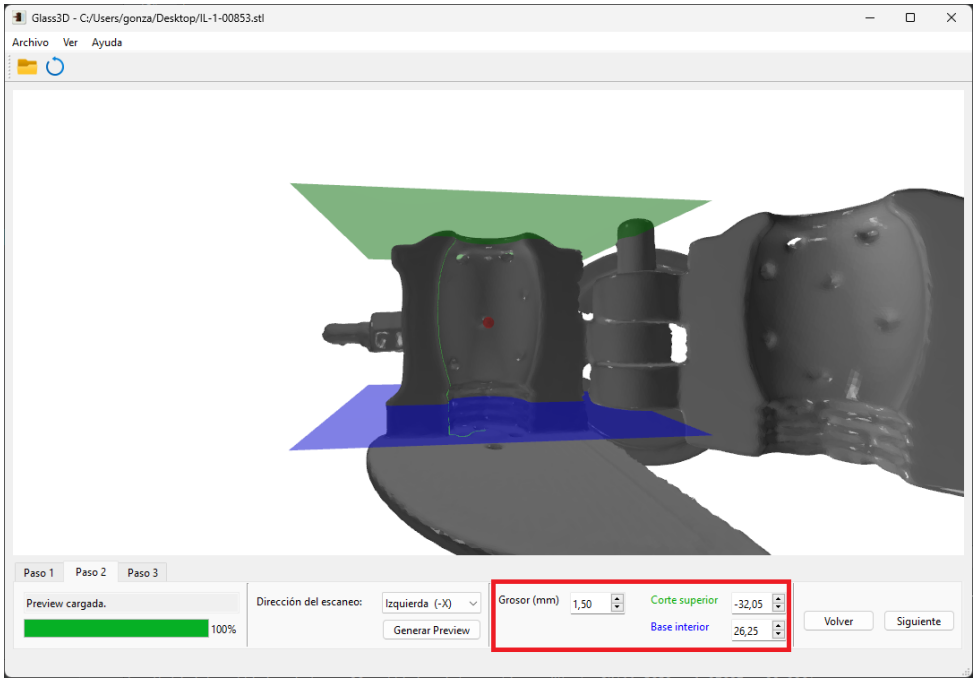

Figura 9.5: Generar Positivo: Ajustar generación

9.1.5. Finalizar proceso

Esta etapa final tiene el objetivo de guardar el modelo resultante. Pasos:

1. Comprobar el resultado mostrando u ocultando el positivo o el molde con los checkbox del paso 3. Figura 9.6.

2. Pulsar el botón "Guardar".

3. Seleccionar la ruta de destino y aceptar.

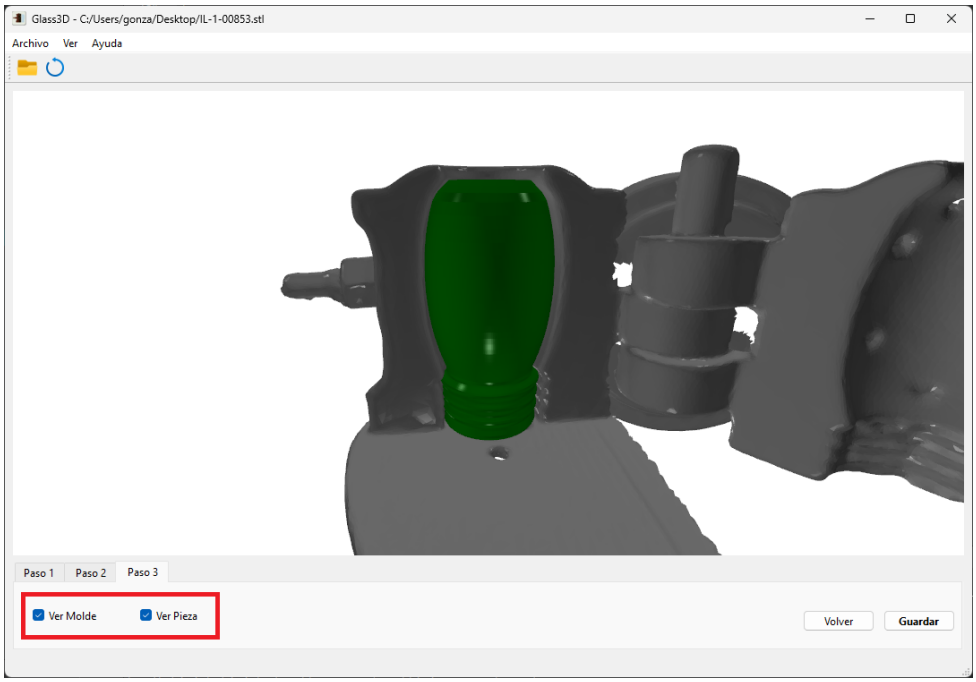

Figura 9.6: Generar Positivo: Comprobar resultado

Para reiniciar el proceso se debe abrir un nuevo modelo desde la opción del menú superior.

9.2. Manual Avanzado

Este manual muestra igualmente el funcionamiento básico de la aplicación, añadiendo explicaciones del funcionamiento de algunas funciones. Además se incluyen recomendaciones de uso y modificación de preferencias. Aunque el documento desde el que se accede en la aplicación incluye todo el contenido del manual básico sumado al nuevo, en esta sección se incluirán solo las diferencias para facilitar la lectura.

9.2.1. Preferencias

Las preferencias de la aplicación se pueden modificar desde la opción de menú **Ver** → **Preferencias**. Los valores ajustables se dividen en dos grupos. Valores por defecto y valores de generación. Ver figura 9.7.

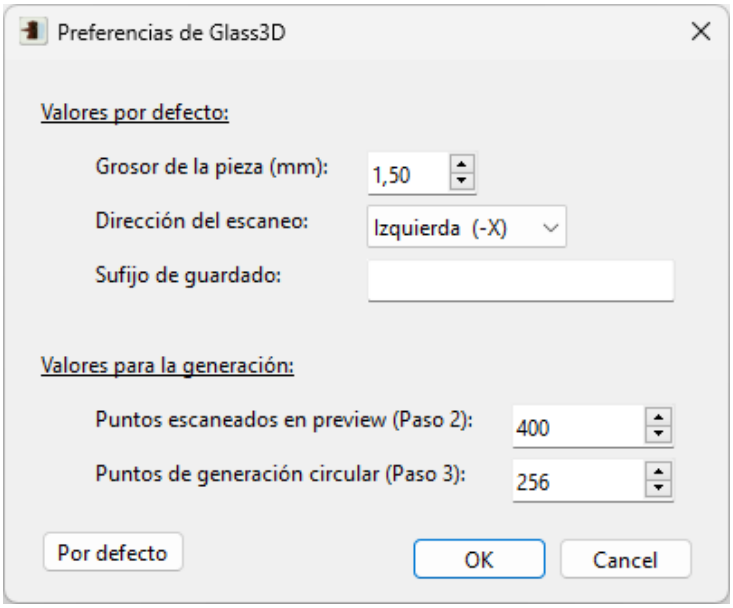

Figura 9.7: Preferencias de Glass3D

Los valores por defecto están destinados a proporcionar comodidad al usuario evitando que tenga que modificar siempre estos valores en cada generación. Son los siguientes:

- Grosor de la pieza (mm): Grosor de las paredes del positivo a generar. Mínimo 0.05 mm.
- Dirección del escaneo: Dirección en la que se escanea el contorno de un molde al generar la preview. Izquierda (-X) o Derecha (+X).
- Sufijo de guardado: Texto que se añade al nombre del molde en el momento de guardar el resultado. Para agilizar la exportación.

La modificación de estos valores influye directamente en la calidad del resultado final y el tiempo de procesado. Más puntos generarán un mejor resultado pero requerirá más tiempo. Los valores de generación son:

- Puntos escaneados en preview: Número de puntos que se detectarán en el momento de definir el contorno sobre el molde. Mínimo 50, máximo 2000.
- Puntos de generación circular: Número de puntos que tendrá cada circunferencia creada a partir de un punto escaneado. Mínimo 20, máximo 500.

9.2.2. Iniciar el proceso

La aplicación no cargará nada en caso de que se intente importar un archivo STL corrupto. Esto podría deberse a fallos en la composición del archivo o a una modificación forzosa de formato. Por ejemplo, cambiar la extensión manualmente de un archivo .obj a .stl resultaría en la misma situación.

Los dos iconos que se muestran en la barra de herramientas son las opciones más usadas:

- Archivo → Abrir.
- Ver → Reiniciar: Esta opción devuelve la cámara a su posición inicial. Es una funcionalidad muy usada y su atajo de teclado es Ctrl + R.

9.2.3. Paso 1: Definir el centro

La definición del centro se realiza detectando dos colisiones desde el punto seleccionado por el usuario. Estas colisiones están en direcciones opuestas del eje X, como se puede ver en la figura 9.8. La profundidad y altura del punto seleccionado se mantienen. Teniendo esto en cuenta se recomienda:

- No seleccionar puntos fuera del interior del molde. Se mostrará un mensaje de aviso cuando no se puedan obtener las dos colisiones comentadas anteriormente desde un punto pero, si existe la posibilidad el centro se genera igualmente. Aunque es muy probable que lo haga en una zona no adecuada para los pasos posteriores.
- No definir el centro en una zona demasiado profunda del molde. Esto producirá un escaneo incorrecto que resultará posteriormente en un positivo con poca fidelidad al molde.
- Si el molde tiene zonas con más detalle se recomienda establecer el centro a una próxima a éstas. Así los detalles se escanearán con mayor precisión.

Figura 9.8: Sistema de centrado

9.2.4. Paso 2: Generar Preview

El escaneo del contorno se hace mediante la emisión de rayos desde el centro en todas las direcciones necesarias (Figura 9.9). Detectando el total de puntos definido en las preferencias. Teniendo en cuenta este funcionamiento, se recomienda desplazar el centro volviendo al paso anterior si el contorno escaneado no es el esperado.

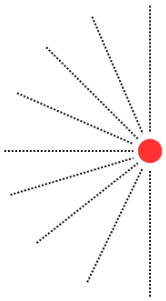

Figura 9.9: Sistema de escaneo del contorno

9.2.5. Paso 3: Generar Positivo

La generación de un positivo no fallará en caso de que los planos de corte y base interior se intercambien o salgan de la superficie del molde. De igual forma no se producirá un error si el grosor de la pieza es mayor al diámetro de la misma. La modificación descontrolada de estos parámetros generará positivos poco fieles a su molde importado.

9.2.6. Finalizar proceso

El nombre de archivo usado por defecto en el momento del guardado será el del propio molde importado. Añadiendo, si se ha definido alguno, el sufijo establecido en las preferencias. El nombre de guardado siempre será también modificable de forma manual.

Parte IV

Apéndices

Apéndice A

Anexos

A.1. Diagramas

Figura A.1: DC: Clases MainWindow y Settings

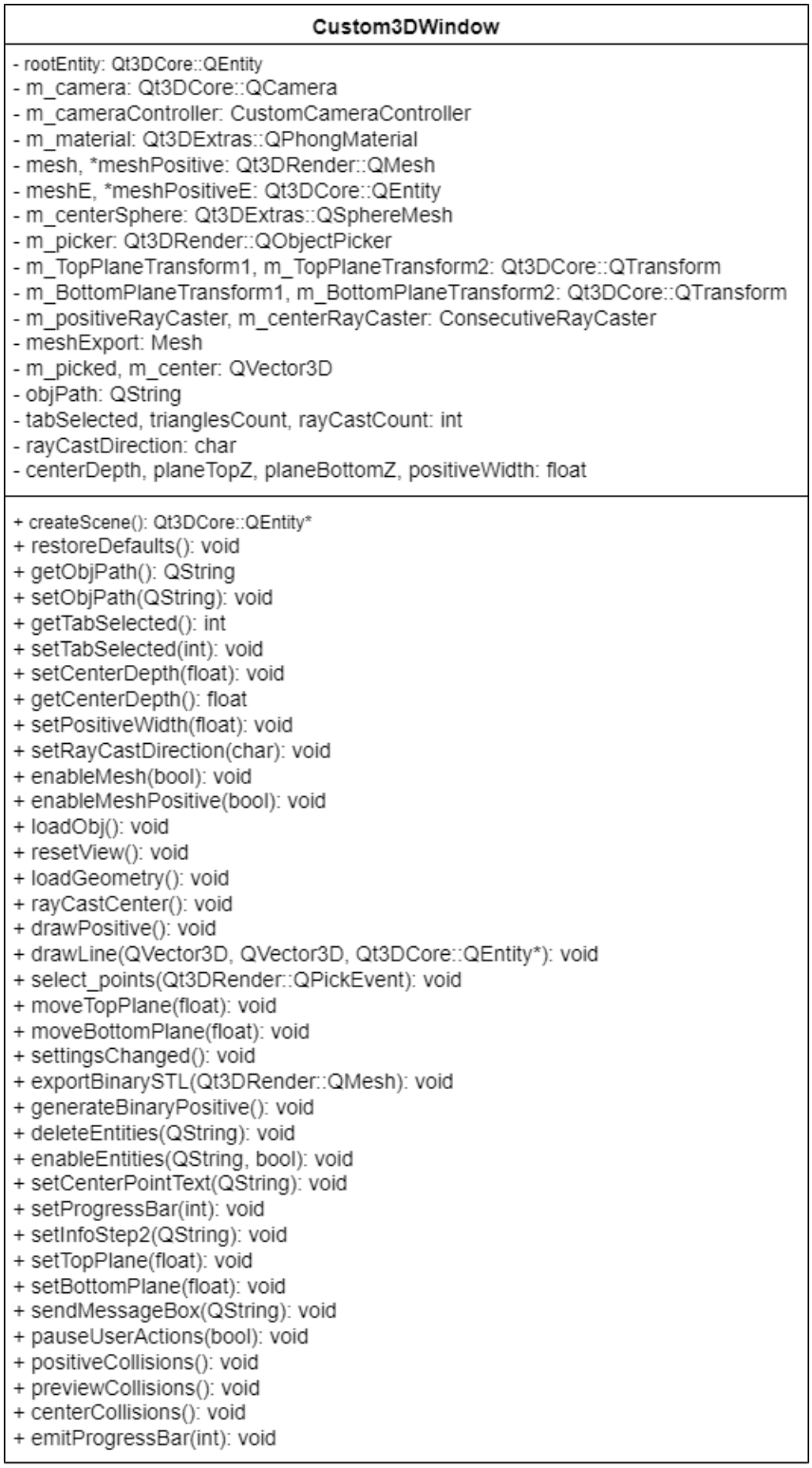

Figura A.2: DC: Clase Custom3DWindow

Figura A.3: DC: Clase CustomCameraController

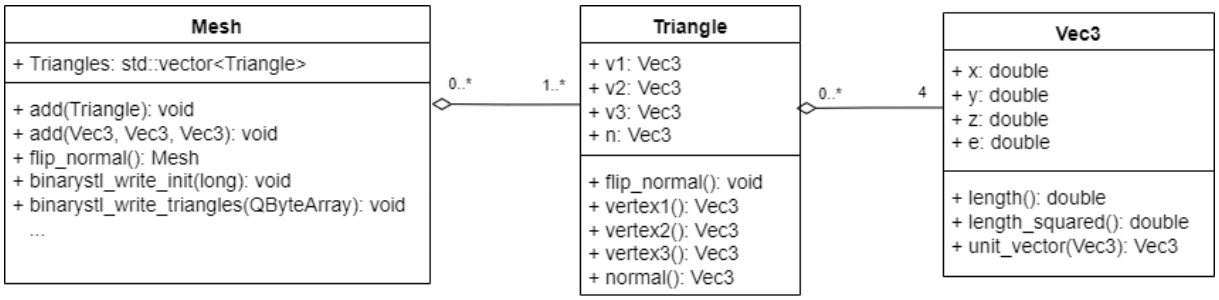

Figura A.4: DC: Clases Mesh, Triangle y Vec3

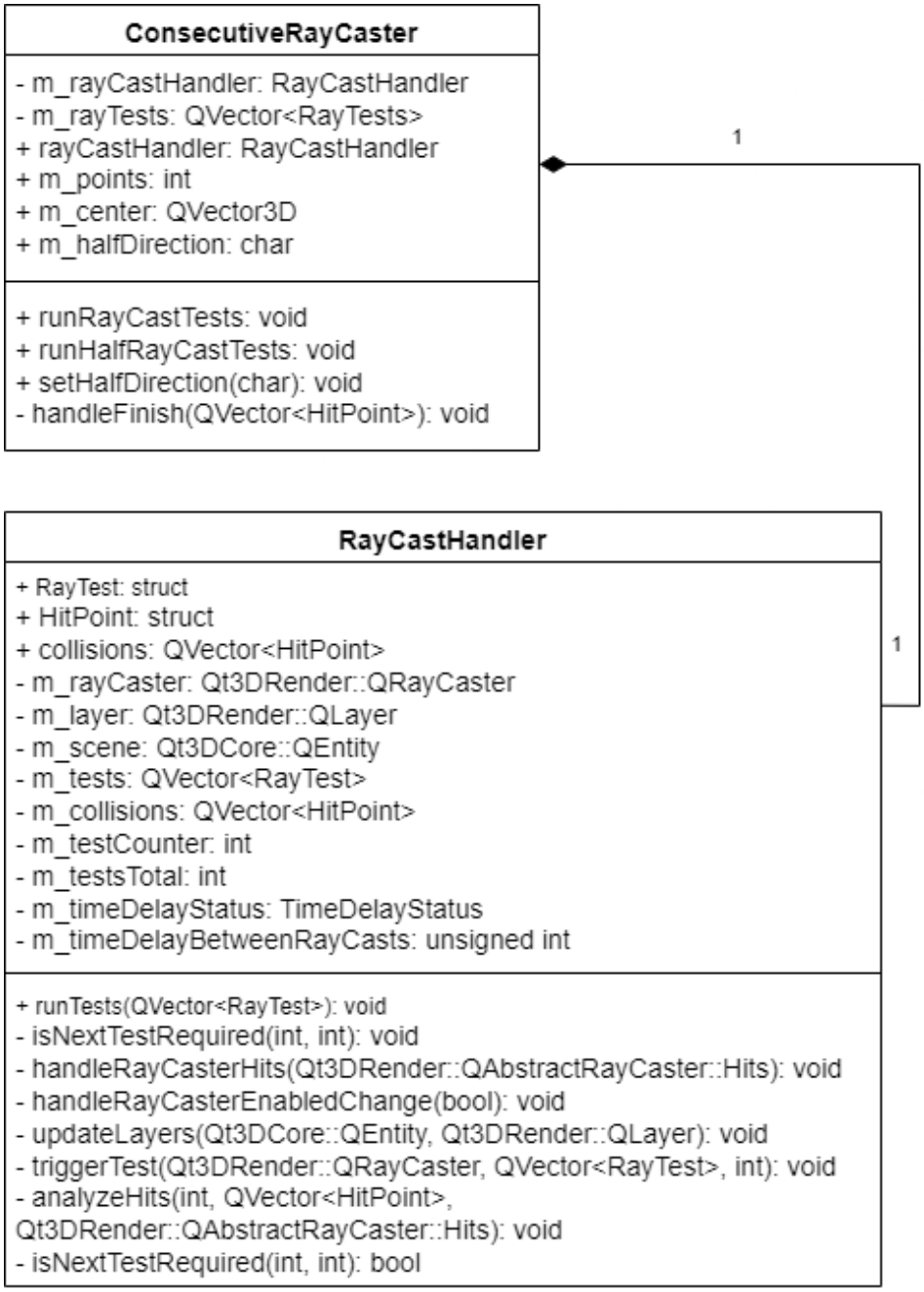

Figura A.5: DC: Clases ConsecutiveRayCaster y RayCastHandler

Webgrafía

[1] Florian Blume. *Setting up ray-caster*. Última vez visitado: 02/05/2023. stackoverflow. URL: https://stackoverflow.com/questions/54900335/signal-to-slot-connection-triggering-signal-iteratively-inside-a-loop/54901850#54901850.

[2] *Canva*. Última vez visitado: 18/06/2023. URL: https://www.canva.com/.

[3] *CodeBrowser*. Última vez visitado: 07/05/2023. URL: https://codebrowser.dev/.

[4] *code.qt.io - examples*. Última vez visitado: 21/04/2023. Qt Company. URL: https://code.qt.io/cgit/qt/qt3d.git/tree/examples/qt3d?h=5.6.

[5] *Documentación oficial de Qt3D*. Última vez visitado: 12/06/2023. Qt Company. URL: https://doc.qt.io/qt-6/qt3d-index.html.

[6] *draw.io*. Última vez visitado: 26/06/2023. URL: https://app.diagrams.net/.

[7] elerac. *stl-creator*. Última vez visitado: 18/05/2023. GitHub. URL: https://github.com/elerac/stl-creator/tree/74f66783faf7be95c9afe8fd7abe2d4ffae77018.

[8] *Overleaf*. Última vez visitado: 28/06/2023. URL: https://www.overleaf.com/.

[9] Dmitrii Tikhonkikh. *Qt3D CPP Tutorial 2. Custom camera controller*. Última vez visitado: 31/03/2023. YouTube. URL: https://www.youtube.com/watch?v=dz1AV_6DF6E.

[10] DuarteCorporation Tutoriales. *Formación en C++*. Última vez visitado: 16/03/2023. YouTube. URL: https://www.youtube.com/@DuarteCorporationTutoriales.

[11] user3405291. *Trigger ray cast tests consecutively*. Última vez visitado: 02/05/2023. stackoverflow. URL: https://stackoverflow.com/questions/57995843/trigger-ray-cast-tests-consecutively.

[12] vre user3405291. *Understanding the mesh created by Qt3D*. Última vez visitado: 12/05/2023. stackoverflow. URL: https://stackoverflow.com/questions/52039817/understanding-the-mesh-created-by-qt3d.

[13] *Using Qt3D. Ring 1.11*. Última vez visitado: 21/04/2023. URL: https://ring-lang.github.io/doc1.11/qt3d.html.

[14] *Wikipedia*. Última vez visitado: 26/06/2023. URL: https://es.wikipedia.org/wiki/Wikipedia:Portada.