Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén

Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

PROTOTIPO PARA LA GENERACIÓN DE OBJETOS 3D CON TOPOLOGÍA IMPLÍCITA

Alumna

María de la Paz Barbero Rodríguez

Tutor

Francisco de Asís Conde Rodríguez
(Departamento de Informática)

septiembre, 2019

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Francisco de Asís Conde Rodríguez, tutor del Trabajo Fin de Grado titulado: '**Prototipo para la generación de objetos 3D con topología implícita**', que presenta Doña María de la Paz Barbero Rodríguez, otorga el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, septiembre de 2019

La alumna: El tutor:

María de la Paz Barbero Rodríguez Francisco de Asís Conde Rodríguez

(Página intencionalmente en blanco)

Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sistemas Gráficos
Idioma	Español
Tipo	Específico
TFT en equipo	No
Fecha de asignación	12/11/2018
Descripción corta	Existen diversos medios para representar objetos 3D con superficies curvadas complejas que no requieren explicitar su topología en una estructura de datos, sino que las relaciones entre los vértices que conforman los distintos triángulos de la malla se calculan matemáticamente. Estos medios son ideales para asignaturas de programación de aplicaciones gráficas, ya que permiten a los estudiantes contar con modelos de objetos con formas complejas sobre los que probar shaders avanzados, sin necesidad de programar las estructuras de datos que los almacenan. De esta forma se pueden concentrar en los aspectos de programación de los shaders. En la asignatura de Programación de Aplicaciones Gráficas se explica uno de esos medios, los objetos de revolución. En este TFG se propone realizar una herramienta para que los estudiantes puedan experimentar con otros medios como lofting. El resultado de este TFG se usará en la asignatura Programación de Aplicaciones Gráficas.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

Contenido

Índice de ilustraciones

Índice de tablas

1 - ESPECIFICACIÓN DEL PROYECTO

En este capítulo se presenta la especificación del proyecto, con una estructura y contenidos que siguen los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

1.1 - Índice general

Como índice de la documentación requerida por la norma UNE 157801 se utilizará en su lugar la tabla de contenido global del presente TFT, ya que todos los documentos indicados en dicha norma están contenidos en el mismo documento maestro.

1.2 - Memoria

A continuación, se presentan los contenidos de la especificación del proyecto definido en el presente TFT, y en el que se describen todos los elementos que deberán entregarse al final de la ejecución del proyecto, así como la especificación de los métodos, herramientas y métricas utilizadas para garantizar el éxito del mismo.

1.2.1 - Introducción

El objetivo del proyecto es el tanto el aprendizaje de nuevas técnicas para la creación de modelos 3D como la observación de las cualidades que presentan estos. Todo ello mediante la creación de una aplicación que genera objetos tridimensionales con curvas complejas a partir de unos datos espaciales que introduce el usuario.

Para ello, la aplicación permite visualizar los objetos generados desde diferentes perspectivas y mostrando algunas propiedades. Además cabe la posibilidad de guardarlos en un fichero.

1.2.2 - Objeto del proyecto

Los objetivos que pretende alcanzar el proyecto son:

- Determinar objetos con formas curvadas complejas con la ayuda de algoritmos que calculan de forma implícita la geometría y la topología. Es el objetivo principal, la obtención de modelos 3D mediante el uso de curvas de paramétricas cúbicas.
- Visualizar los objetos que se han creado como resultado del punto anterior. Este objetivo es fundamental para poder comprobar si los parámetros introducidos para la creación han sido correctos y se ha obtenido el resultado esperado por el usuario.
- Modificar parámetros de los objetos y ver el efecto que se obtiene. A partir de los primeros datos introducidos en la aplicación se pueden hacer modificaciones sobre estos para crear nuevos modelos diferentes a los anteriores. Para observar los cambios que se han producido al cambiar los parámetros introducidos previamente.
- Salvar a disco los objetos definidos. Debe ser posible almacenar los puntos y demás información (topología, normales y coordenadas de textura) obtenidos en la creación del objeto para tener la posibilidad de inspeccionar los datos en el fichero y poder experimentar en otros programas los datos generados.

1.2.3 - Antecedentes

Durante las prácticas de la asignatura Programación de Aplicaciones Gráficas se desarrolló una aplicación similar que generaba modelos 3D mediante perfiles de revolución. Aunque al principio me pareció algo complejo, los resultados obtenidos fueron lo suficientemente satisfactorios como para saber que quería realizar algo similar como trabajo de fin de grado. Por ello, mi tutor me propuso realizar una aplicación muy similar en la que se incluyera el manejo de curvas para la generación de modelos. En este caso la aplicación tenía que incluir una interfaz gráfica que la dotara de usabilidad y comodidad. Además, creo que para otros compañeros que se vayan a iniciar en la materia puede ser una herramienta para su aprendizaje.

1.2.4 - Descripción de la situación actual

En el ámbito del modelado 3D se pueden encontrar numerosas aplicaciones con las que obtener resultados similares a los que ofrece la aplicación. Este es el caso de Blender, un potente software open source para creación 3D. Sin embargo, todas ellas presentan un alto grado de complejidad debido a las numerosas opciones que ofrecen.

1.2.4.1 - Descripción del entorno actual

Este proyecto se podría definir dentro de un entorno relacionado con el ámbito de la educación, sirviendo de apoyo a los estudiantes que estén comenzando con el aprendizaje de técnicas de modelado. Por ejemplo, en la asignatura de Programación de Aplicaciones Gráficas se podría estudiar como anexo a la técnica del perfil de revolución. Además, para estudiantes de otras titulaciones también podría ser de ayuda, en el estudio de CAD (diseño asistido por ordenador), donde se utilizan primitivas similares a las que la aplicación puede generar.

1.2.4.2 - Resumen de las principales deficiencias identificadas

La principal deficiencia que se puede apreciar en la mayoría del software que permite crear modelos parecidos a los que se obtienen en la aplicación es la dificultad. Cuando un usuario principiante abre por primera vez alguno de estos programas se puede ver desbordado ante la cantidad de información que se le presenta.

Una persona que se está iniciando en el modelado no está habituada al uso de estos programas con extensos menús y decenas de opciones, lo que hace que para obtener el resultado deseado tenga que navegar por internet para averiguar cómo hacerlo.

Este hecho me ayudó a comprender que en la aplicación habría que mostrar en la interfaz lo estrictamente necesario para no abrumar a un usuario que no está acostumbrado a ningún software similar. De esta forma, alguien sin muchos conocimientos puede obtener un buen resultado en pocos instantes, cosa que, con otros programas se puede ver frustrada.

Como consecuencia, podrá visualizar en pocos segundos un modelo y comprobar si es el resultado que esperaba de forma intuitiva y con una interfaz muy sencilla y con pocos elementos.

1.2.5 - Normas y referencias

A continuación, se presentan las normas, reglamentos y referencias de cualquier tipo que han sido de aplicación en la elaboración del proyecto o en la ejecución del mismo. La norma UNE 157801 incluye aquí un apartado sobre bibliografía. Sin embargo, al integrarse el documento de especificación del proyecto en el documento global del TFG, la bibliografía de este capítulo se integra junto con la bibliografía del documento maestro.

1.2.5.1 - Métodos, herramientas, modelos, métricas y prototipos

Las herramientas que se han utilizado en el desarrollo de la aplicación son las siguientes:

- Como IDE se ha utilizado Qt Creator 4.9.0 (Community) en su versión gratuita.
- Para crear la interfaz gráfica de usuario se ha usado la biblioteca Qt 5.12.3 con la ayuda del software mencionado en el punto anterior.
- OpenGL 4.1.0 Core Profile como biblioteca para gestionar la parte gráfica de la aplicación.
- C++ 11 como lenguaje de programación.
- GLSL como lenguaje para programar los shaders.
- Visual Paradigm como herramienta de ingeniería del software.

1.2.5.2 - Mecanismos de control de calidad aplicados durante la redacción del proyecto

El aseguramiento de la calidad en la redacción del proyecto implica la verificación de la completitud (falta de omisiones), la integridad de la documentación del proyecto, así como una redacción clara, concisa y entendible por todos los participantes e interesados en el proyecto. Además, deben establecerse mecanismos de verificación de la integridad y completitud de la documentación.

Al estar el proyecto desarrollado por un solo autor y verificado por un tutor, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, el autor del proyecto ha utilizado mecanismos básicos para la verificación de la integridad y completitud de la documentación del proyecto,

que incluyen un control de versiones a nivel de documentación y un control sencillo de la trazabilidad de requisitos y especificaciones del proyecto.

1.2.6 - Definiciones y abreviaturas

Para la correcta comprensión del proyecto es necesario el conocimiento de una serie de términos. Estos son:

- Punto/vértice: es un trío de coordenadas (x,y,z) que representa una localización en el espacio 3D. Se representará con una letra mayúscula (normalmente la P).
- Vector: es un trío de coordenadas (x,y,z) que representa una dirección en el espacio 3D. Se representará como una letra minúscula en negrita.
- Geometría: con este término se hace referencia al conjunto de puntos en el espacio 3D que forman parte de un objeto o modelo 3D.
- Topología: es la forma en la que los puntos de la geometría se organizan y relacionan entre sí para formar el modelo.
- Curva paramétrica cúbica: define los puntos que compone una curva 3D mediante tres polinomios (uno para cada coordenada x, y, z) en función de un parámetro *t* que pertenece al intervalo [0,1]. Para representarla se utilizan polinomios de grado 3 para que se pueda controlar la curva con mayor flexibilidad y facilidad.
- Continuidad paramétrica de orden 0 o C^0: este grado de continuidad implica que los extremos de la curva se toquen. En otras palabras, que los valores x, y, z de la curva c_1 cuando t=1 sean los mismos que los valores de la curva c_2 cuando t=0.
- Continuidad paramétrica de orden 1 o C^1: en este caso, para que se cumpla esta continuidad, la primera derivada (representa la tangente de la curva) en el punto de conexión de las dos curvas es igual en ambas secciones de la curva. La magnitud del vector tangente debe ser la misma para ambas curvas. La continuidad C^1 implica la continuidad G^1, pero no al contrario.

- Continuidad geométrica de orden 0 o G^0: tiene las mismas condiciones que la continuidad C^0.
- Continuidad geométrica de orden 1 o G^1: este tipo de continuidad implica que las tangentes que tiene el punto de conexión tengan la misma dirección, pero no necesariamente el mismo módulo. De esta forma, ambos vectores tangentes son proporcionales.
- Curva de Bézier: curva paramétrica cúbica desarrollada por Pierre Bézier en la que las tangentes en los extremos vienen determinadas por dos puntos que no pertenecen a la curva. Los puntos P_1, P_2, P_3 y P_4 se denominan puntos de control o coeficientes geométricos. Estas curvas interpolan los extremos, es decir, pasan por los extremos (P_1 y P_4 son los puntos inicial y final respectivamente). Además, aproximan los otros dos puntos de control (P_2 y P_3 no pertenecen a la curva pero influyen en su forma), ya que las tangentes (que indican la dirección de la curva) se calculan con los vectores que forman los puntos P_1P_2 y P_3P_4.

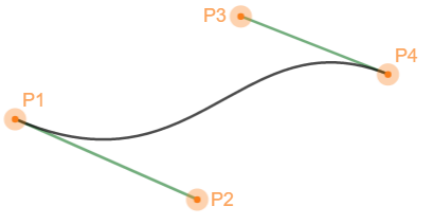


Ilustración 1.1. Ejemplo de curva de Bézier

- Funciones mezcla de Bézier: son cuatro ecuaciones ($B_0^3, B_1^3, B_2^3, B_3^3$) que definen una curva de Bézier. Ponderan los cuatro puntos generadores en función de un parámetro t.

$$B_0^3(t) = (1-t)^3 \qquad B_1^3(t) = 3t(1-t)^2$$
$$B_2^3(t) = 3t^2(1-t) \qquad B_3^3(t) = t^3$$

Representadas gráficamente:

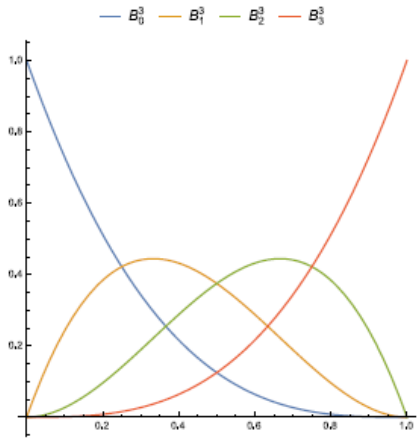


Figura 6.4: Funciones mezcla de Bézier. *Generadas con Mathematica 10*

Ilustración 1.2. Funciones mezcla de Bézier representadas gráficamente. Conde, F. (2015). Modelado de sólidos heterogéneos mediante hiperparches.

- Perfil de barrido/ puntos del perfil: es un conjunto de puntos que se definen en un plano 2D y que se van trasladando a lo largo de una trayectoria para formar la malla de triángulos que compone al modelo.
- Subdivisión de perfil: técnica usada para suavizar las líneas que forman el conjunto de puntos del perfil. Consiste en generar nuevos puntos partiendo de los iniciales mediante una media ponderada, obteniendo así un conjunto con un mayor número de vértices más próximos entre sí. De esta forma, la superficie resultante no presentará formas tan abruptas.
- Tramo/ tramo de Bézier: normalmente las curvas se suelen dividir en segmentos o trozos para que estos sean más manejables. Un tramo de Bézier pertenece a la ruta total. Cada uno de ellos está formado por cuatro puntos de control que definen la curva de Bézier.
- Path/ruta: es la curva resultante de unir todos los tramos de Bézier que la componen. Es el “camino” que deben seguir los puntos del perfil para formar la geometría del modelo 3D.
- Superficie: cuando se habla se superficie se hace referencia al modelo que se ha generado barriendo los puntos del perfil a través de cada tramo de la ruta.

- Shader (Shader program): programa que está diseñado para ejecutarse en alguna etapa del pipeline de rendering en la GPU. Nos permiten indicar a la GPU cómo queremos que se dibuje.
- Vector normal: vector ortogonal a todos los vectores tangentes de una entidad geométrica.
- Vector tangente: vector que es tangente a una superficie o curva en un punto. En las curvas de Bézier, las tangentes indican la dirección de la curva.
- Coordenadas de textura: estas coordenadas (u,v) se calculan para cada vértice de la geometría e indican una correspondencia con las coordenadas en la textura. Los valores de u y v están normalizados en el intervalo [0,1].
- Rotación CCW: en este ámbito, este concepto hace referencia al modo de construir la topología de la malla de triángulos. Los vértices que forman un triángulo deben ser nombrados en sentido contrario a las agujas del reloj, para que, de esta manera, las normales de los triángulos sean correctas y coherentes al modelo.
- VAO o Vertex Array Object: es un objeto de OpenGL que almacena datos de los vértices que se van a dibujar en la GPU. Contiene información sobre el formato de los datos además de “Buffer Objects”, que contienen arrays con los datos de los vértices.
- VBO o Vertex Buffer Object: es un objeto de OpenGL que almacena los datos correspondientes a la geometría del modelo. En nuestro caso, almacenan las posiciones, normales y coordenadas de textura de cada uno de los vértices.
- IBO o Index Buffer Object: es un objeto de OpenGL que almacena información correspondiente a la topología del modelo. En otras palabras, almacena las relaciones que hay entre los vértices mediante sus índices.
- Triangle strip: es una primitiva de OpenGL que se usa a la hora de dibujar para indicar que se va a dibujar una tira de triángulos. Así, cuando la

GPU recibe los índices, sabe cómo tiene que relacionarlos para formar los triángulos.

- Singleton: es un patrón de diseño que se basa en crear solamente una instancia de una clase para poder acceder siempre de la misma manera a ella.

1.2.7 - Requisitos iniciales

Los requisitos que podemos deducir de la aplicación son:

- En primer lugar, el sistema debe permitir la creación de un modelo 3D a partir de unos datos de entrada. Este es el requisito principal, ya que es en torno a lo que gira toda la aplicación.
- Para que se pueda generar el objeto 3D, el sistema debe permitir que se definan los puntos que van a dar forma a este. Por una parte, debe ser capaz de recopilar los puntos para definir el perfil y, por otra parte, los que definen la ruta.
- A la hora de definir la ruta, el sistema debe tener en cuenta los tramos en los que se divide la curva, ofreciendo así la posibilidad de crear curvas complejas.
- Para hacer los modelos más complejos, el sistema debe ofrecer al usuario la posibilidad de generar el modelo con mayor detalle en el perfil y en la curva.
- Una vez se haya creado el objeto, el sistema debe ofrecer una visualización del resultado, para que el usuario compruebe si era lo que esperaba o tiene que modificar algún parámetro.
- En relación al punto anterior, el sistema debe exportar el modelo generado en un fichero con un formato adecuado.
- Todo lo anterior debe ser visualizado en la interfaz gráfica de usuario, permitiendo así un mayor grado de interactividad con el mismo.

1.2.8 - Alcance

Al finalizar el proyecto, se podrá disponer de estos elementos:

- Un ejecutable. La aplicación con la que el usuario interactúa.
- La memoria del proyecto (este documento).
- Un manual de usuario. Para facilitar el aprendizaje del uso de la aplicación.

1.2.9 - Hipótesis y restricciones

El TFT se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

1.2.10 - Estudio de alternativas y viabilidad

En primer lugar, se estudió de entre una serie de alternativas la biblioteca a usar para crear interfaces gráficas y que fuera compatible con OpenGL. De este estudio se dedujo que la biblioteca idónea para realizar la aplicación sería Qt, ya que además de proporcionar numerosas facilidades a la hora de diseñar y crear interfaces, es una tecnología robusta: presenta actualizaciones cada uno o dos años y tiene una comunidad extensa y activa en los foros.

La elección de Qt como biblioteca desembocó en el uso del propio IDE de Qt, Qt Creator. Es un entorno sencillo que ofrece la posibilidad de construir las interfaces gráficas de forma “visual”, sin necesidad de hacerlo mediante código.

En cuanto al lenguaje de programación, se eligió C++ porque es uno de los lenguajes en los que más se ha trabajado a lo largo de las asignaturas del grado. Además, a la hora de gestionar datos que van directamente a la GPU, C++ permite un manejo sencillo de estos.

En el ámbito matemático, había que decidir cómo se iban a representar y trabajar las curvas que se necesitan para la creación de los modelos. Finalmente se optó por las curvas paramétricas cúbicas ya que son las curvas más sencillas que a su vez permiten formas más complejas. Dentro de este grupo, había varias alternativas, las curvas de Hermite en las que se tiene en cuenta los puntos inicial y final y los vectores tangentes en esos puntos, y las curvas de Bézier, en las que se toma como referencia cuatro puntos de control. Una ventaja que proporcionan las curvas de Bézier es que interpolan los extremos, lo que hace que se tenga aún más

control sobre la curva y los resultados que se esperan dados los puntos generadores. En este caso, se optó por las curvas de Bézier ya que para un usuario principiante es más intuitivo y sencillo proporcionar cuatro puntos antes que proporcionar dos puntos y calcular dos tangentes.

En cuanto a viabilidad, este proyecto puede ser concebido desde el punto de vista de una organización como una herramienta gratuita para dar a conocer la organización. Por este motivo, el proyecto es viable en cuanto a prestigio podría aportar a la empresa. Sin embargo, al ser una herramienta que se va a distribuir sin coste, no habría beneficios económicos.

1.2.11 - Descripción de la solución propuesta

Al término de la implementación de la aplicación, esta podrá:

- Ofrecer al usuario un espacio en la interfaz para introducir los puntos con los que se va a formar el perfil de barrido.
- Permitir que el usuario introduzca las veces que se realizará la subdivisión del perfil de barrido.
- Del mismo modo, se podrá añadir los puntos que van a definir la curva de la trayectoria o ruta.
- El usuario también podrá especificar la precisión con la que se construya la curva definida por los puntos anteriores.
- Cuando el modelo se haya creado, poder visualizar el resultado en la interfaz gráfica.
- Guardar la información de geometría, topología, coordenadas de textura y normales del objeto generado.
- Visualizar la información del modelo generado gracias al uso de diferentes shaders.

En esta aplicación no se observan riesgos destacables.

1.2.12 - Planificación temporal

Se ha previsto que el proyecto se pueda implementar en dos meses de duración. Se ha usado la metodología clásica de desarrollo por lo que las principales etapas vendrán marcadas por esta. Hay que señalar que el número de días indicado son días laborables. Así, podemos señalar estas fases del proyecto:

- La primera fase es el análisis. Al principio es necesario saber qué se tiene que hacer. 5 días.
- A continuación, se realiza una etapa de investigación en la que se estudian las posibilidades que ofrecen las diferentes tecnologías necesarias para cubrir los requisitos del proyecto. 5 días.
- El siguiente paso es el diseño. En él se da forma a la estructura del código. 10 días
- A continuación, se procede a la implementación del diseño. Esta etapa debe ir acompañada del estudio matemático de la solución. Dentro de esta fase se pueden señalar distintas tareas: 25 días
    1. Realizar modificaciones en el código para pasarlo al entorno de Qt.
    2. Implementar las clases que se encargan de construir los modelos.
    3. Implementar la interfaz.
    4. Conectar las partes nuevas al código ya existente.
    5. Realizar pruebas para comprobar el funcionamiento.
- Por último, habría que realizar las pruebas para comprobar que se han cumplido los requisitos. 3 días
- Durante todo este proceso hay que ir confeccionando a la par una documentación que recoja los aspectos que se han ido realizando.

1.2.13 - Resumen del presupuesto

El total del coste de la realización de este proyecto asciende a 5100 euros. Más adelante se detallan los puntos que dan lugar a esta cifra.

1.2.14 - Orden de prioridad de los documentos básicos del proyecto

El proyecto se presenta dentro de este capítulo del documento global con los contenidos indicados en la norma UNE 157801. El orden de prioridad utilizado es el siguiente:

1. Memoria del proyecto.
2. Especificaciones del sistema.
3. Presupuesto.
4. Estudios con entidad propia.
5. Anexos según la norma (numerados como A1...AN).

1.3 - Especificaciones del sistema

Los requisitos funcionales de la aplicación son:

- En primer lugar, el sistema debe permitir la creación de un modelo 3D a partir de unos datos de entrada. En otras palabras, el sistema debe generar un conjunto de vértices con sus normales y coordenadas de textura correspondientes y generar las relaciones entre los vértices.
- Para que se pueda generar el objeto 3D, el sistema debe permitir que se definan los puntos que van a dar forma a este. Para esta tarea debe ofrecer un modo para que el usuario pueda introducir los puntos que van a formar parte del perfil y, por otra parte, los puntos que van a definir la curva de la trayectoria.
- En caso en el que el número de datos suministrados por el usuario sea insuficiente, el sistema no debe realizar nada.
- Para que el usuario pueda crear una trayectoria de barrido más compleja, la ruta debe estar dividida en tramos. Así cada tramo tendrá una curva correspondiente. Para ello, se debe dar la posibilidad al usuario de crear nuevos tramos o trozos que se añadan a los existentes y formen un todo con ellos.
- Una vez se haya creado el objeto, el sistema debe ofrecer una visualización del resultado, para que el usuario compruebe si era lo que

esperaba o tiene que modificar algún parámetro. Esta visualización puede venir de la mano de una ventana de visualización donde se dibuje el modelo creado.

- En relación al punto anterior, el sistema debe exportar el modelo generado en un fichero con un formato adecuado. De modo que, se pueda tener un archivo con toda la información de la geometría y topología que se han creado.
- Mostrar una interfaz gráfica en la que se disponga de toda la información que el usuario necesita para entender la aplicación y el modelo generado.

Los requisitos no funcionales son:

- La interfaz debe ser usable, ya que lo que la aplicación pretende es resolver una de las principales deficiencias encontradas en programas similares: ser intuitivo.
- El programa debe tener un tiempo de respuesta aceptable, de forma que los modelos no tarden en generarse y visualizarse.

1.4 - Presupuesto

El presupuesto para este proyecto se ha calculado en base a:

Debido al grado de complejidad del proyecto, se ha estimado que bastaría con un empleado para encargarse de él. Por tanto, solo sería necesaria la adquisición y configuración de un solo equipo.

En cuanto a la adquisición de los equipos se refiere, se ha tenido en cuenta un costo medio de entre los equipos que cumplen los requisitos para desarrollar esta aplicación (entre 800 € y 1200€). Por otra parte, se ha estimado que la vida útil del equipo es de seis años.

Por otra parte, para calcular el sueldo de los empleados se ha considerado un sueldo para un programador junior (unos 1500 €).

Los costes indirectos derivados de los anteriores se calcularán como el 20% de la suma de los otros costes. Esto hace un total de 1100 €.

Cabe destacar que, debido a que la aplicación no se va a comercializar, interesa escoger la versión Open Source de Qt, ya que es gratuita. Además, OpenGL no va a generar costes ya que es gratuita.

Concepto	Duración	Coste unitario	Unidades	Total
Adquisición de los equipos	6 años	1000€	1	1000 €
Sueldo de los empleados	2 meses	1500€	1	3000 €
Uso de Qt	2 meses	0 €	1	0 €
Costes indirectos				1100 €

Tabla 1.1. Cálculo de presupuesto

En este caso, se podría amortizar el coste del equipo, ya que es lo único que forma parte del activo no corriente de la empresa (en términos de este proyecto). Además, se puede estimar que el valor residual del ordenador sea de 150 €.

La base amortizable es 850 € (precio de adquisición - valor residual).

Realizando una amortización por suma de dígitos creciente:

- La cuota de amortización total es 40,48 € (base amortizable/ suma nº de años[1])
- En el año 1 la cuota de amortización es 40,48€
- En el año 2: 80,96€
- En el año 3: 121,44€
- En el año 4: 161,92€
- En el año 5: 202,4€
- En el año 6: 242,88€

[1] Esta operación se realiza con la duración del proyecto. Al ser esta tan breve, se ha sustituido por el número de años de la vida útil, suponiendo que la organización mantenga el equipo tras finalizar el proyecto.

1.5 - A1: Documentación de entrada

Como equivalente al pliego de condiciones se incluye la documentación de entrada especificada en el Apéndice Guía original del Trabajo Fin de Título.

1.6 - A2: Análisis y Diseño del sistema

Análisis:

El usuario debe introducir los datos para generar el modelo, tanto los puntos del perfil de barrido como los puntos para dar forma a la curva. Además, el usuario debe indicar si quiere que se suavicen las formas mediante subdivisión del perfil y/o de las curvas.

Debido a que una curva cúbica de Bézier admite solo cuatro puntos de control y los modelos que se quieren construir pueden tener mayor complejidad, es conveniente que se descomponga la curva en trozos o tramos. De esta forma, una curva estará compuesta por una o más curvas. Cada una de ellas tendrá cuatro puntos generadores característicos y se generarán todas de forma similar.

En relación con los párrafos anteriores, el sistema tiene que encargarse de procesar los datos introducidos por el usuario y construir el modelo. Para que el modelo sea un todo y no sean curvas separadas en el espacio, los tramos deben unirse y cumplir la continuidad paramétrica de orden 1 o C^1. Así, no se generará ningún tipo de artefacto. Para ello en el primer tramo el usuario tendrá que introducir cuatro puntos, pero para los demás casos solo serán necesarios dos para calcular los otros dos en función de los puntos del tramo anterior.

Para que el usuario pueda introducir todos los datos necesarios para la generación del modelo y su posterior visualización de forma sencilla es necesaria una interfaz gráfica. Esta debe contener elementos que permitan gestionar (insertar, modificar y borrar) tanto los puntos que forman el perfil como los que van a formar la ruta. Además, debe mostrar los puntos que ya han sido añadidos, teniendo así una lista ordenada. Por otra parte, es necesario un elemento en el que se visualice el modelo ya generado, una ventana de OpenGL. Todo ello teniendo en cuenta que el diseño de la interfaz debe ser lo más simple y usable posible.

Se debe añadir una opción en la interfaz para guardar los objetos cuando hayan sido generados. Para ello el formato del archivo será obj, ya que la mayoría de software de informática gráfica trabaja con él. Dentro del mismo, se almacenarán

posiciones, normales, coordenadas de textura y la información referente a la topología.

```
# vertices
v 0.880004 3.12 -0.106
v 3.07495 0.925047 -2.99813
...
# texture
vt 0 0
vt 0 1
...
# normals
vn -0.482021 0.482021 -0.73165
vn -0.482021 0.482021 -0.73165
...
# faces
f v2 v1 v4
f v1 v3 v4
...
```

Ilustración 1.3. Ejemplo de archivo en formato obj

Diseño:

Para realizar la aplicación se va a utilizar como base parte de otra aplicación implementada en la asignatura de Programación de Aplicaciones Gráficas. Por este motivo, se va a reutilizar y adaptar gran parte del código ya generado.

Las clases de las que parte la aplicación son:

- La clase renderer: es el “cerebro” de la aplicación. Por una parte, se encarga de dibujar la escena conectando todos los elementos necesarios para ello y por otra es la que intercambia la información entre la parte de la interfaz y la parte del modelado. Contiene los shaders, cámaras y modelos que ayudan a visualizar la escena.
- La clase camera: representa una cámara en el espacio 3D. Se encarga de ofrecernos una visualización de la escena. Gracias a los métodos que incorpora se puede observar el modelo desde diferentes puntos de vista, ya que implementa diferentes movimientos.
- La clase element3d: abstrae el comportamiento de las clases que representan los modelos. Contiene métodos virtuales correspondientes con el dibujado de la geometría.

- La clase subdivisionProfile: es una clase auxiliar que implementa un algoritmo de subdivisión de polilíneas. Con unos puntos de entrada puede suavizar las formas que se crean entre dichos puntos. Como salida se obtiene un perfil con un mayor número de puntos que tiene más precisión que el original. Ayuda a la hora de generar perfiles con curva sin necesidad de añadir muchos puntos.

Para añadir la nueva funcionalidad:

Como se ha indicado antes, la curva en su totalidad se va a representar en tramos o “trozos” más pequeños para poder definir curvas complejas de forma más sencilla. Cada tramo representa una curva cúbica de Bézier con la precisión que el usuario haya especificado. Para cada tramo se debe construir una parte del modelo, la superficie que formarán los nuevos puntos que se generen mediante el barrido de los puntos del perfil por la trayectoria que dicte la curva de cada tramo.

Del párrafo anterior podemos extraer el comportamiento de tres clases. Una de ellas debe representar la totalidad de los tramos que forman la curva y generan el modelo. Un tramo representará la curva de Bézier que se genera en un trozo de la trayectoria. Esta clase dibujará la curva y los puntos correspondientes a un tramo. La clase que se encarga de barrer el perfil a lo largo de la curva del tramo será la clase superficiePerfil. Esta operación se podría encapsular dentro de la clase tramo, pero para hacerlo más sencillo, cada tramo tendrá su superficie correspondiente y en la clase superficiePerfil se creará el modelo y se dibujará la información del mismo.

La superficie del modelo podrá suavizarse en caso de que el usuario lo haya indicado. Entonces se hará uso de la clase subdivisionProfile ya existente encargada de subdividir perfiles.

Por otra parte, es necesaria la creación de una clase llamada mainWindow que maneje el comportamiento de la ventana principal en la que se va a crear la interfaz gráfica. Esta clase debe almacenar la información que se introduce para luego enviarla al renderer y crear el modelo con esos datos.

Otro elemento indispensable en la interfaz es la ventana de OpenGL en la que se va a dibujar. Esta ventana se va a materializar en la clase glwindow y va a manejar

los eventos que se generen sobre la misma. Para incorporarla a la ventana principal debe cumplir el requisito de Qt: heredar de QWidget.

1.6.1 - Metodología de desarrollo

Debido al grado de complejidad del proyecto se ha usado la metodología tradicional compuesta por las fases de análisis, diseño, implementación y pruebas.

1.7 - A3: Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFT, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (el autor).

2 - DESARROLLO DEL PROYECTO

2.1 - Tecnologías utilizadas

Los lenguajes que se han utilizado son:

- C++: lenguaje de propósito general basado en C.
- GLSL (OpenGL Shading Language): lenguaje que nos permite la creación de shader programs.

Las librerías usadas son:

- Qt: biblioteca que nos permite la creación y manejo de interfaces gráficas.
- OpenGL 4.1.0 Core Profile: biblioteca para gestionar la parte gráfica de la aplicación.

2.2 - Diseño

2.2.1 - Diseño arquitectónico del sistema

Como se ha mencionado en apartados anteriores, la aplicación surge como una ampliación del trabajo realizado en las prácticas de Programación de Aplicaciones Gráficas. Por este motivo, el diseño se ha centrado en la nueva funcionalidad de la antigua aplicación. Para ello se han usado el paradigma de la programación orientada a objetos.

El sistema se ha dividido en clases según su funcionalidad, de manera que cada clase tiene una función específica. Así, se ha decidido estructurar el sistema como sigue:

Una clase para gestionar la ruta al completo. Una clase para gestionar cada tramo de la ruta. Además, se ha creado otra clase para generar la geometría de cada tramo de la ruta, o lo que es lo mismo, el modelo en sí.

Por otra parte, se han creado otra clase para manejar los eventos de la interfaz gráfica.

2.2.2 - Diagramas de clases

El diagrama de clases incluyendo todas las clases del proyecto quedaría así:

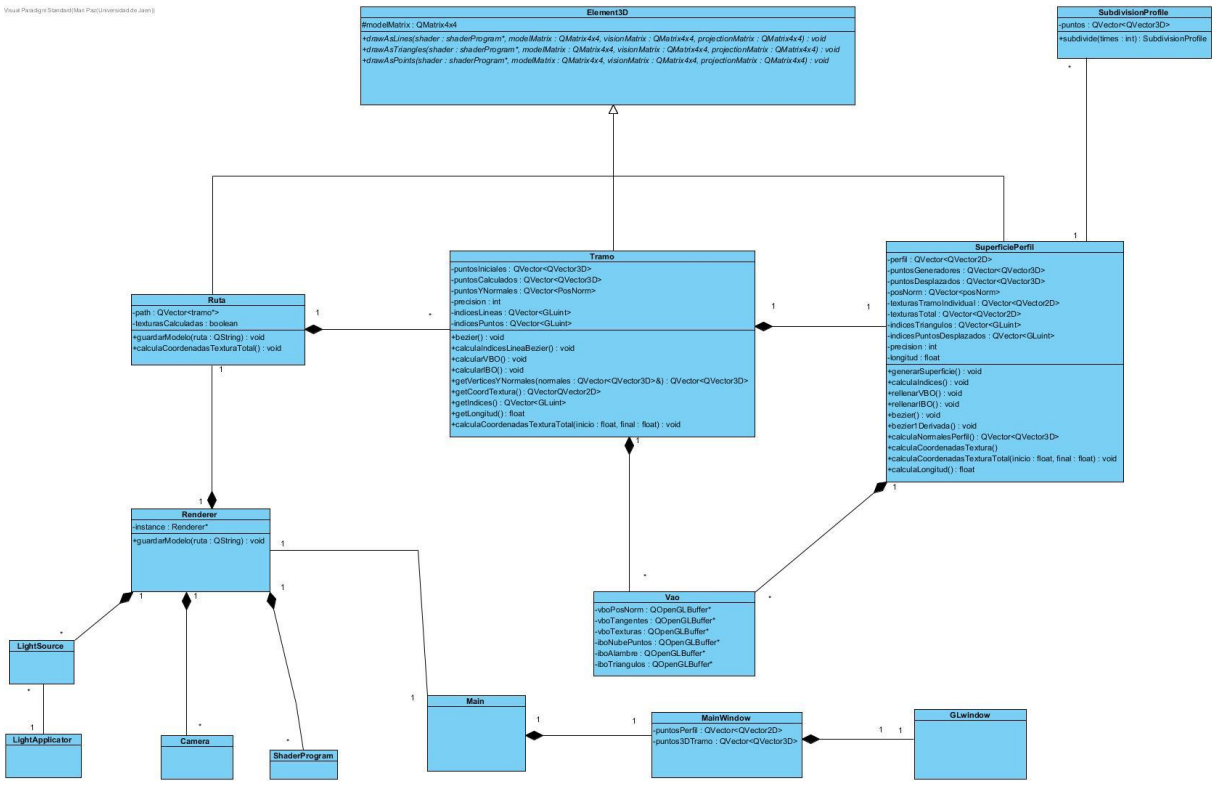


Ilustración 2.1. Diagrama de clases

Cabe destacar que se ha puesto un mayor grado de detalle en las clases nuevas que tienen mayor interés en el objetivo de este trabajo. Las clases que no se han detallado forman parte de la aplicación de partida y, aunque son indispensables, no requieren tanto detalle como las más recientes.

2.2.3 - Diagramas de casos de uso

Caso de uso	Introducir los puntos del perfil
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo usuario
Condición previa	Diseño del perfil y obtención de puntos
Operaciones básicas	
1	Hacer click en “Añadir punto”

2	Introducir coordenadas X e Y
3	Pulsar “Ok”
4	Introducir el número de subdivisiones en el campo “Precisión de los puntos”
Alternativas	
3.1	Si se quiere modificar el punto, seleccionarlo y pulsar “Editar punto”
3.2	Si se quiere eliminar el punto, seleccionarlo y pulsar “Borrar punto”

Tabla 2.1. Caso de uso “Introducir los puntos del perfil”

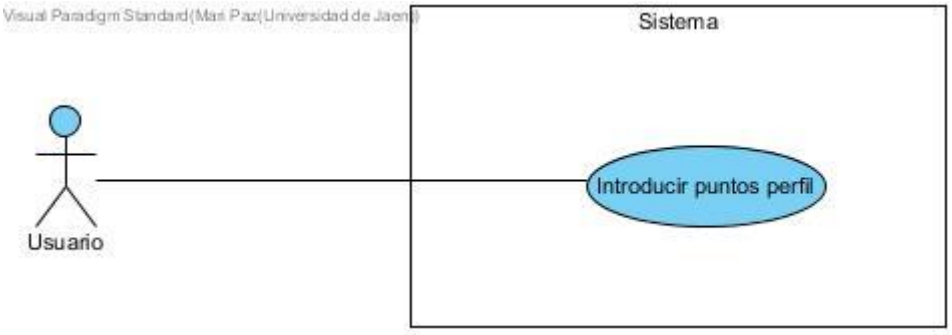


Ilustración 2.1. Caso de uso “Introducir los puntos del perfil”

Caso de uso	Introducir puntos de la ruta
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo usuario
Condición previa	Diseño de la curva
Operaciones básicas	
1	Hacer click en “Añadir tramo”

2	Introducir las coordenadas X,Y y Z de los puntos de la curva
3	Pulsar “Ok”
4	Introducir un número que representa la precisión de la curva en el campo “Precisión de los tramos”
Alternativas	
3.1	Si se quieren modificar los puntos del tramo, seleccionarlo y pulsar “Editar tramo”
3.2	Si se quiere eliminar el tramo, seleccionarlo y pulsar “Borrar punto”

Tabla 1.2. Caso de uso “Introducir puntos de la ruta”

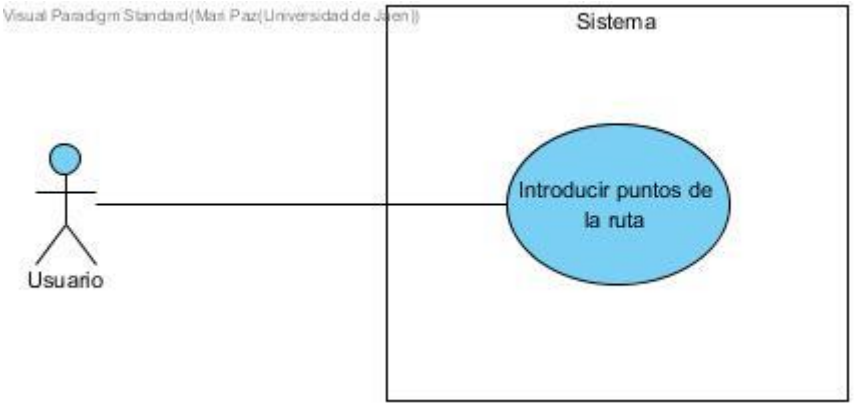


Ilustración 2.2. Caso de uso “Introducir puntos de la ruta”

Caso de uso	Crear modelo
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo usuario
Condición previa	Haber introducido puntos en el perfil y tramos
Operaciones básicas	

1	Pulsar el botón “Crear modelo”

Tabla 2.3. Caso de uso “Crear modelo”

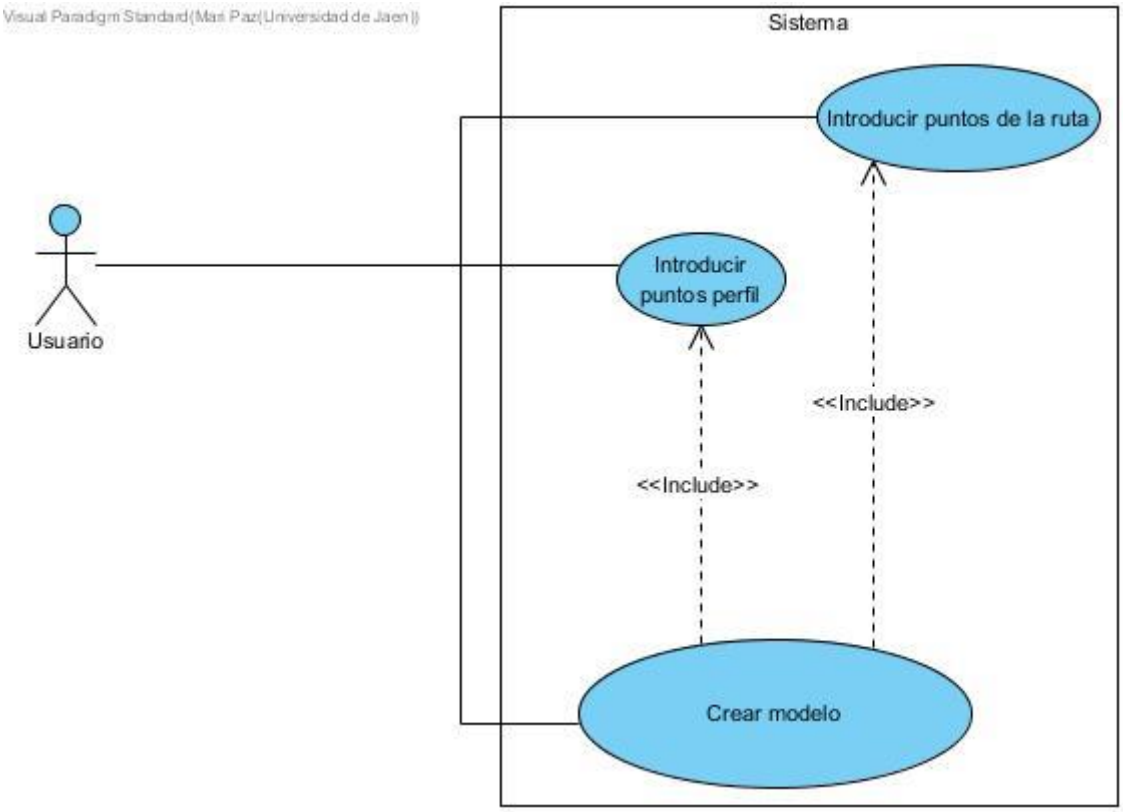


Ilustración 2.3. Caso de uso “Crear modelo”

Caso de uso	Cambiar visualización
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo usuario
Condición previa	Haber creado un modelo
Operaciones básicas	
1	Hacer click en “Ver” situado en la barra superior

2	Seleccionar el modo de visualización del desplegable

Tabla 2.4. Caso de uso "Cambiar visualización"

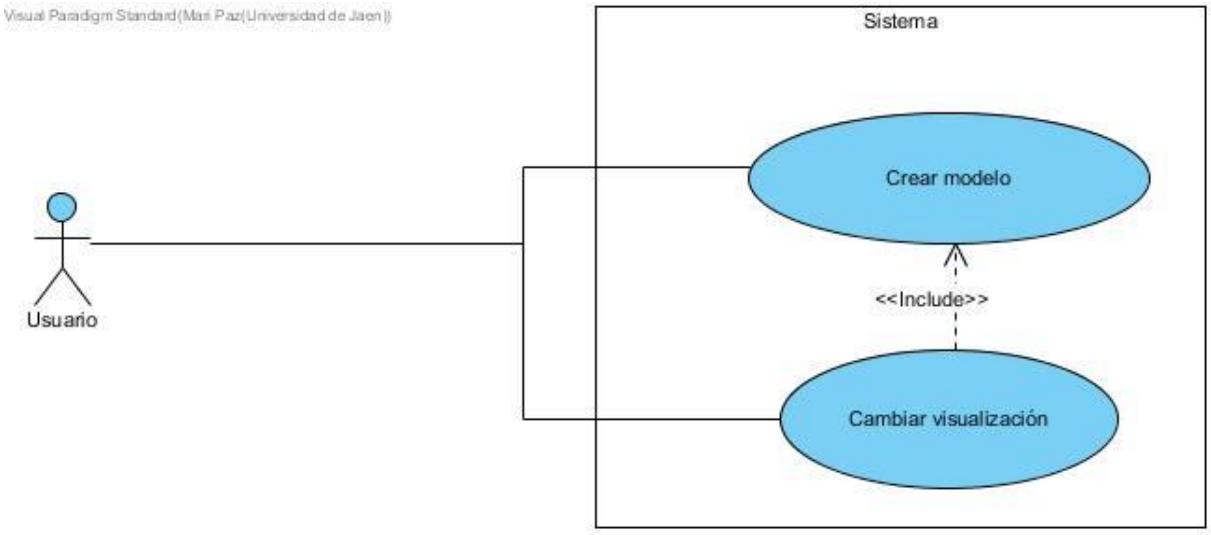


Ilustración 2.4. Caso de uso "Cambiar visualización"

Caso de uso	Guardar modelo
Actor primario	Usuario
Sistema	Aplicación
Participantes	Usuario
Nivel	Objetivo usuario
Condición previa	Haber creado un modelo
Operaciones básicas	
1	Hacer click en "Archivo" situado en la barra superior
2	Pulsar la opción "Guardar modelo"
3	Seleccionar la ruta en la que se quiera guardar
4	Escribir el nombre del archivo que va a almacenar el modelo

5	Pulsar “Guardar”
Alternativas	
5.1	Si no se quiere guardar, pulsar “Cancelar”

Tabla 2.2. Caso de uso “Guardar modelo”

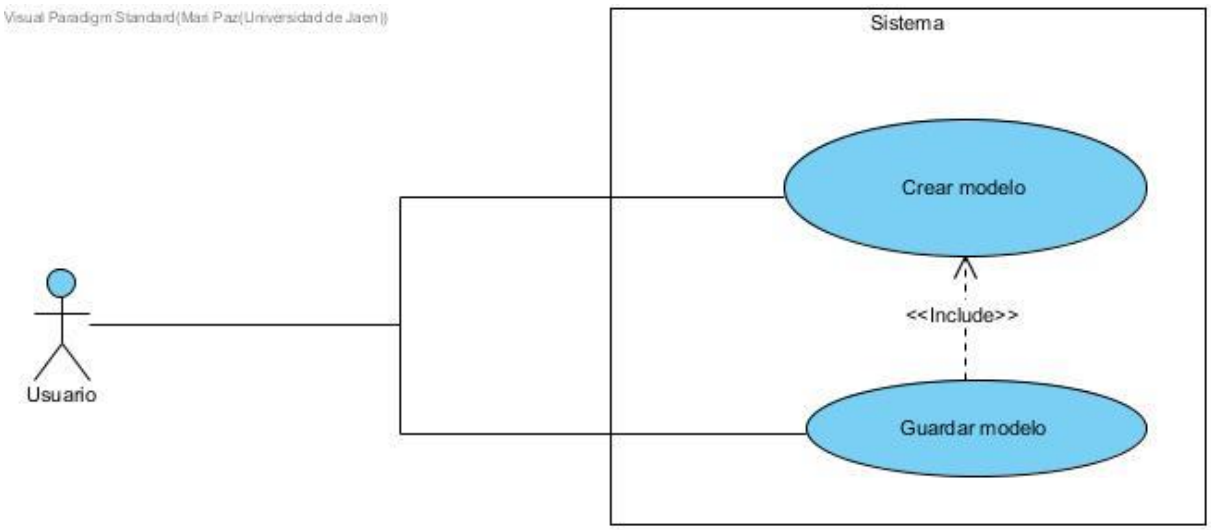


Ilustración 2.5. Caso de uso “Guardar modelo”

2.2.4 - Diagramas de secuencia

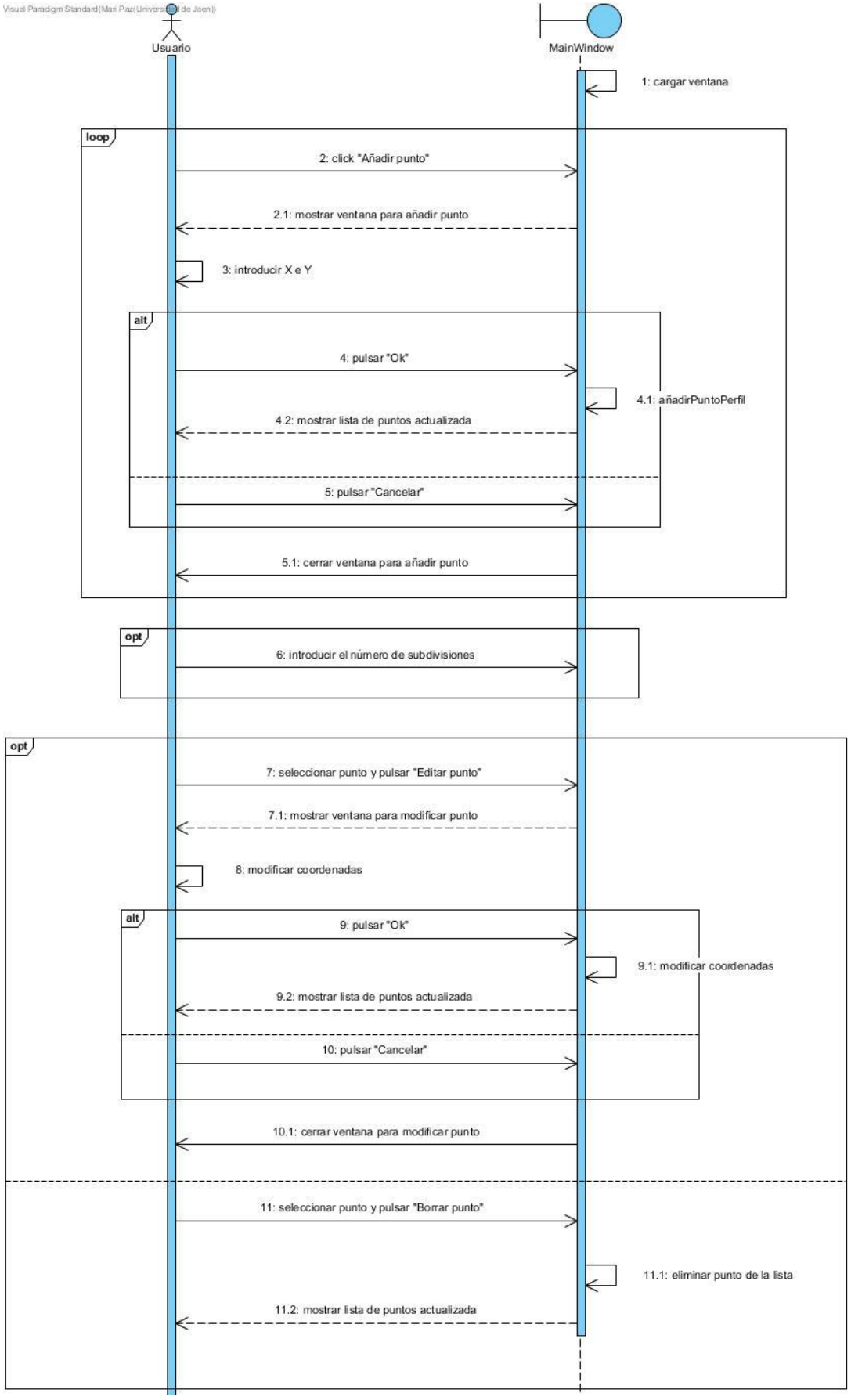


Ilustración 2.6. Diagrama de secuencia "Añadir punto del perfil"

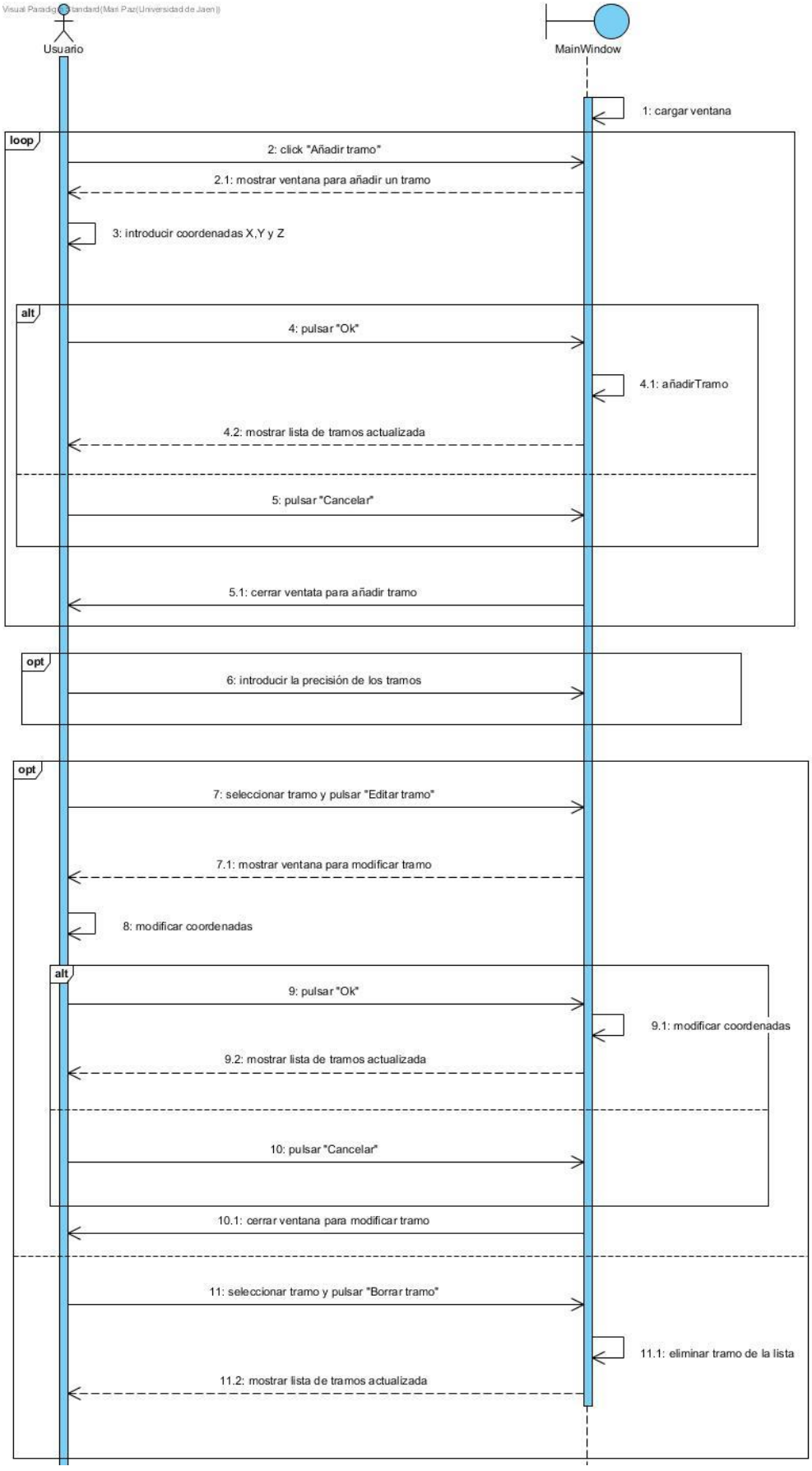


Ilustración 2.7. Diagrama de secuencia "Añadir tramo de la ruta"

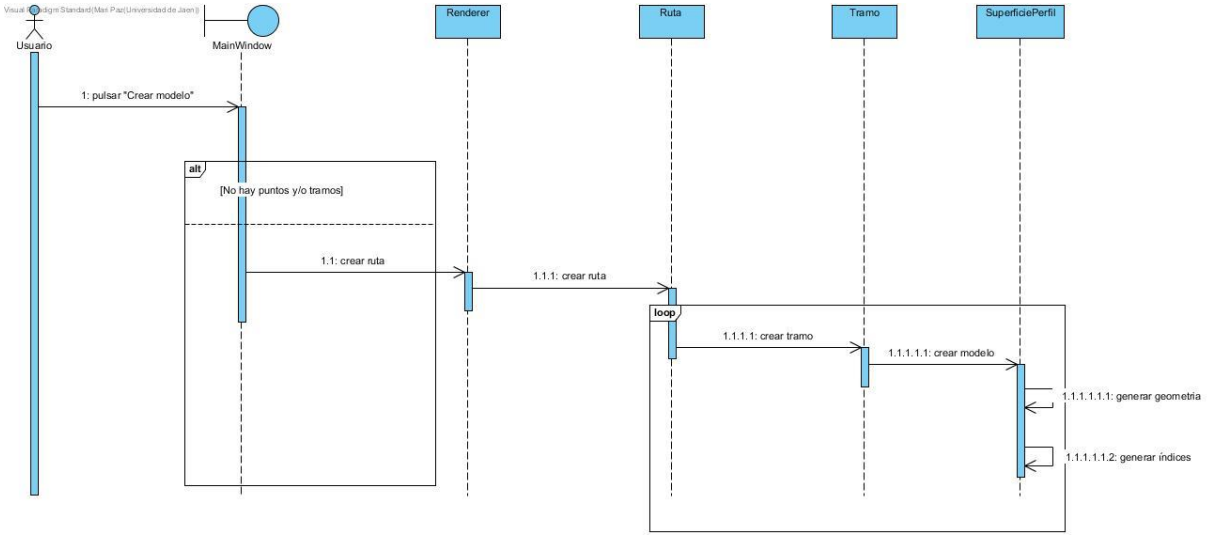


Ilustración 2.8. Diagrama de secuencia "Crear modelo"

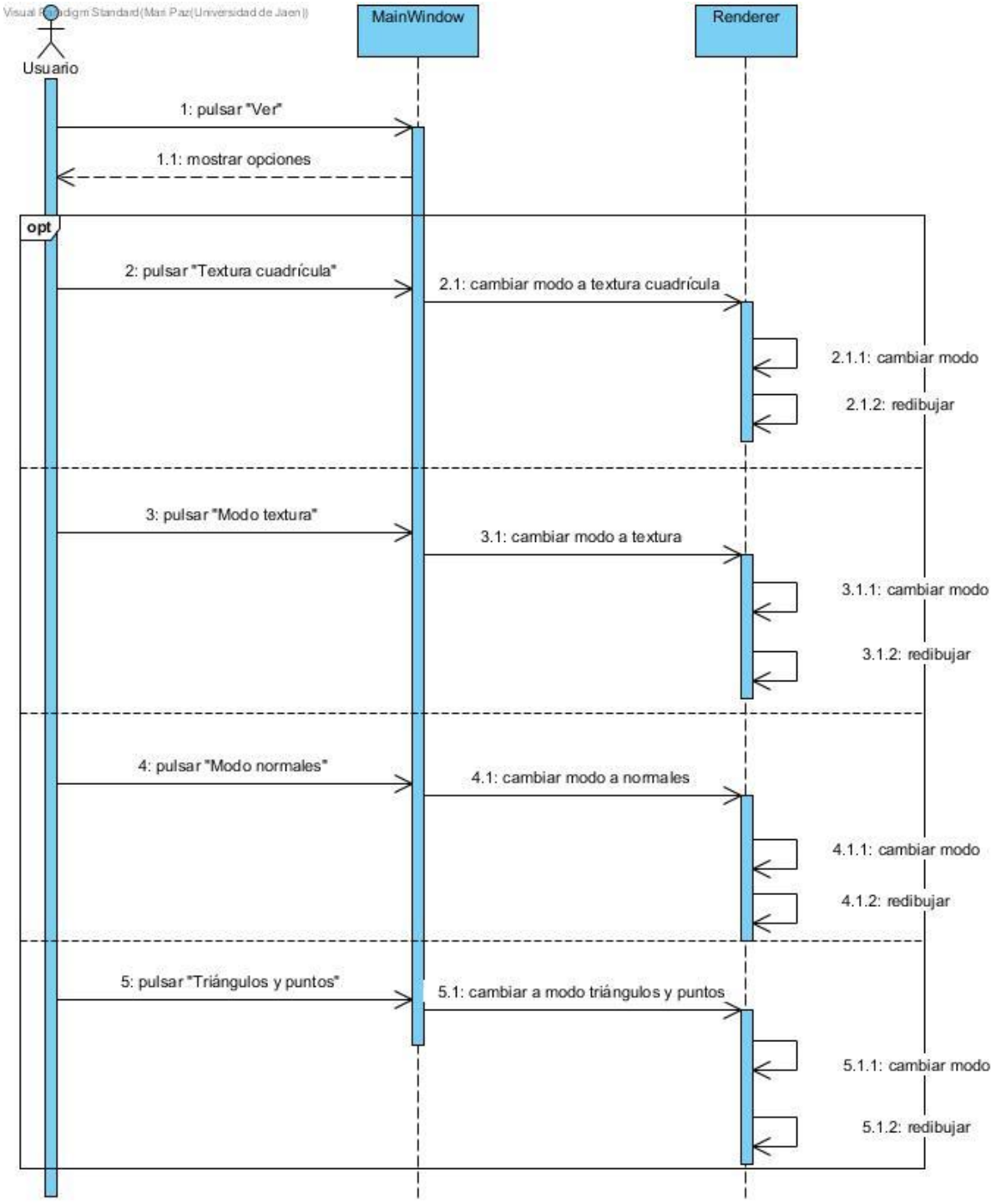


Ilustración 2.9. Diagrama de secuencia "Cambiar modo de visualización"

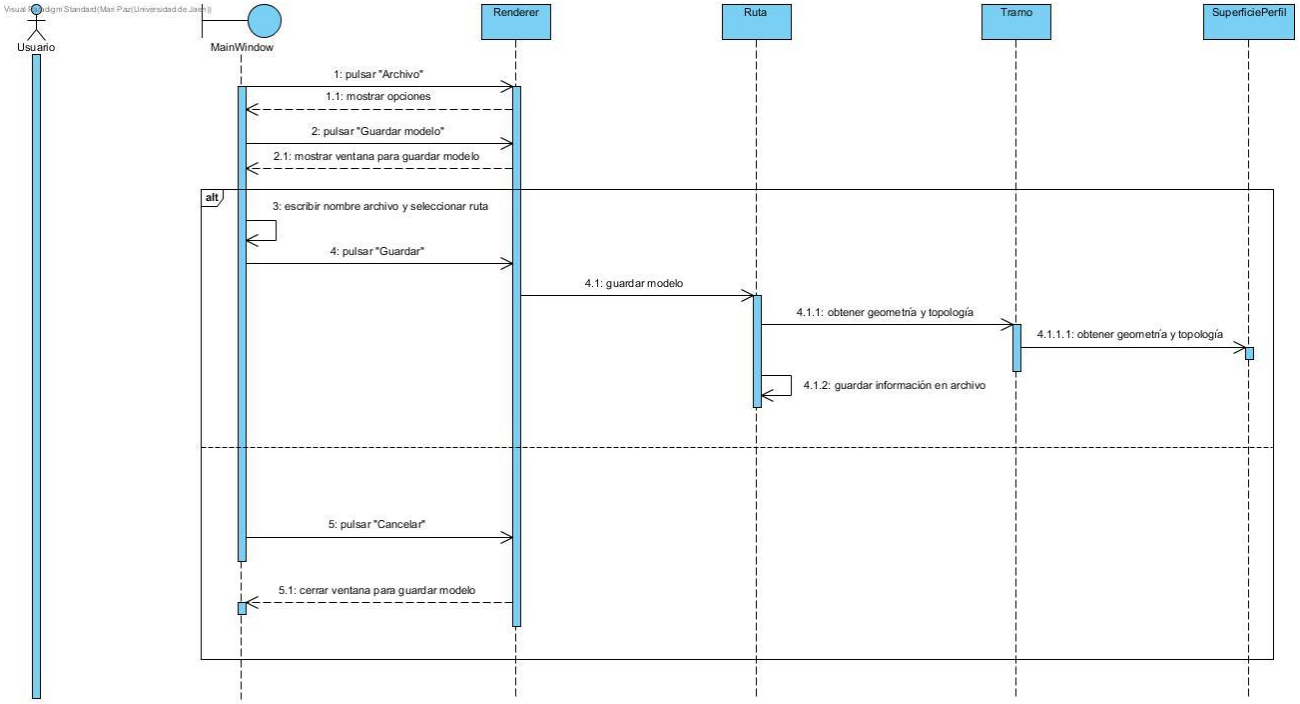


Ilustración 2.10. Diagrama de secuencia "Guardar modelo"

2.2.5 - Diseño de la interfaz y storyboards

El diseño propuesto para la interfaz quedaría como muestra la siguiente imagen.

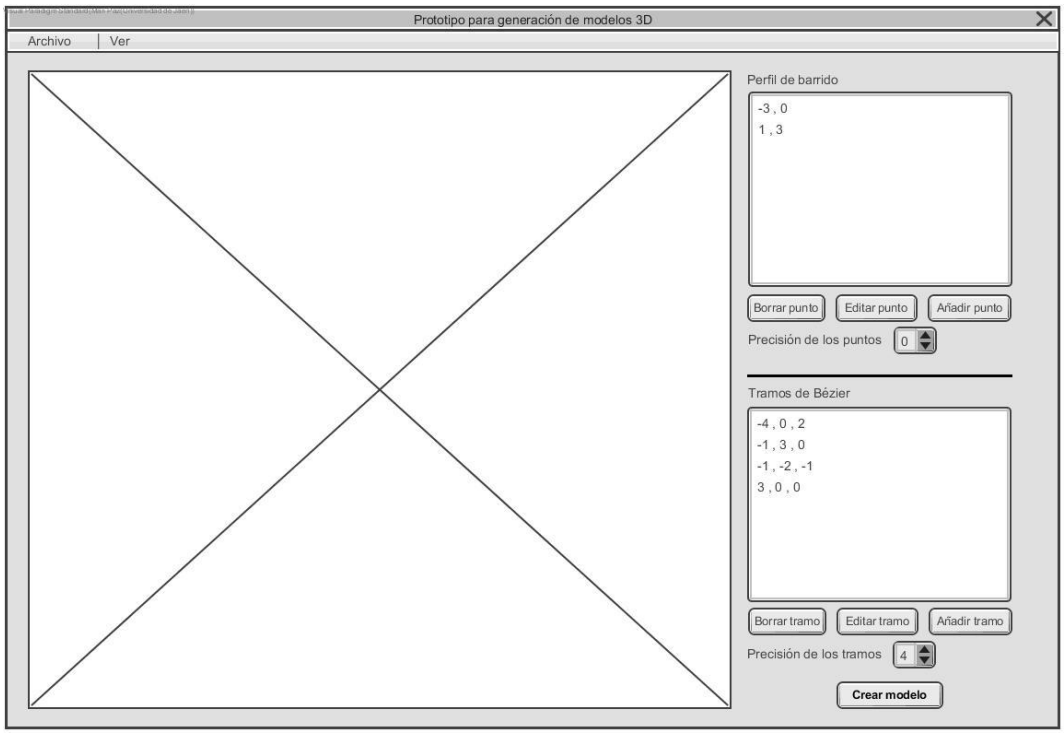


Ilustración 2.11. Interfaz. Ventana principal

La interfaz se compone de una ventana principal en la que en la parte izquierda se muestra el área de dibujo donde se mostrará el modelo que se ha creado.

En la parte derecha encontramos dos listas en las que se añadirá la información para puntos y tramos. Dicha información se podrá gestionar con los tres botones inferiores a cada una de las listas.

Además, para especificar las subdivisiones o la precisión con la que se creará el modelo, se han añadido dos campos numéricos.

Debajo de toda la columna derecha se encuentra el botón que permite crear el modelo una vez introducida la información necesaria.

Para realizar otras acciones se ha colocado una toolbar en la que el usuario podrá acceder a diferentes desplegables para guardar el modelo o cambiar el modo de visualización.

Por otra parte, a la hora de añadir y modificar puntos aparece una ventana en la que se pueden modificar los valores numéricos según cada caso.

El storyboard quedaría así:

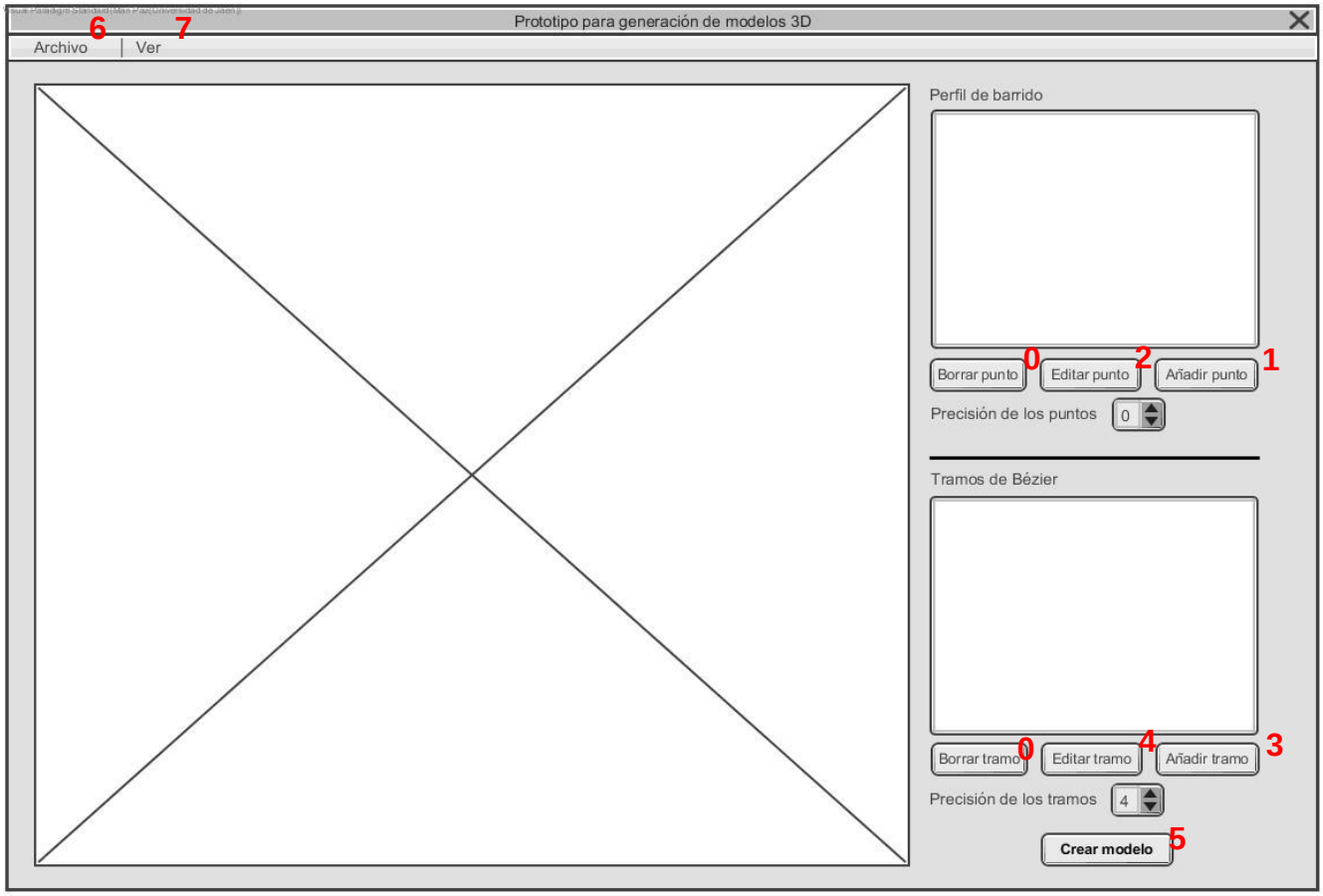


Ilustración 2.12. Storyboard pantalla 0

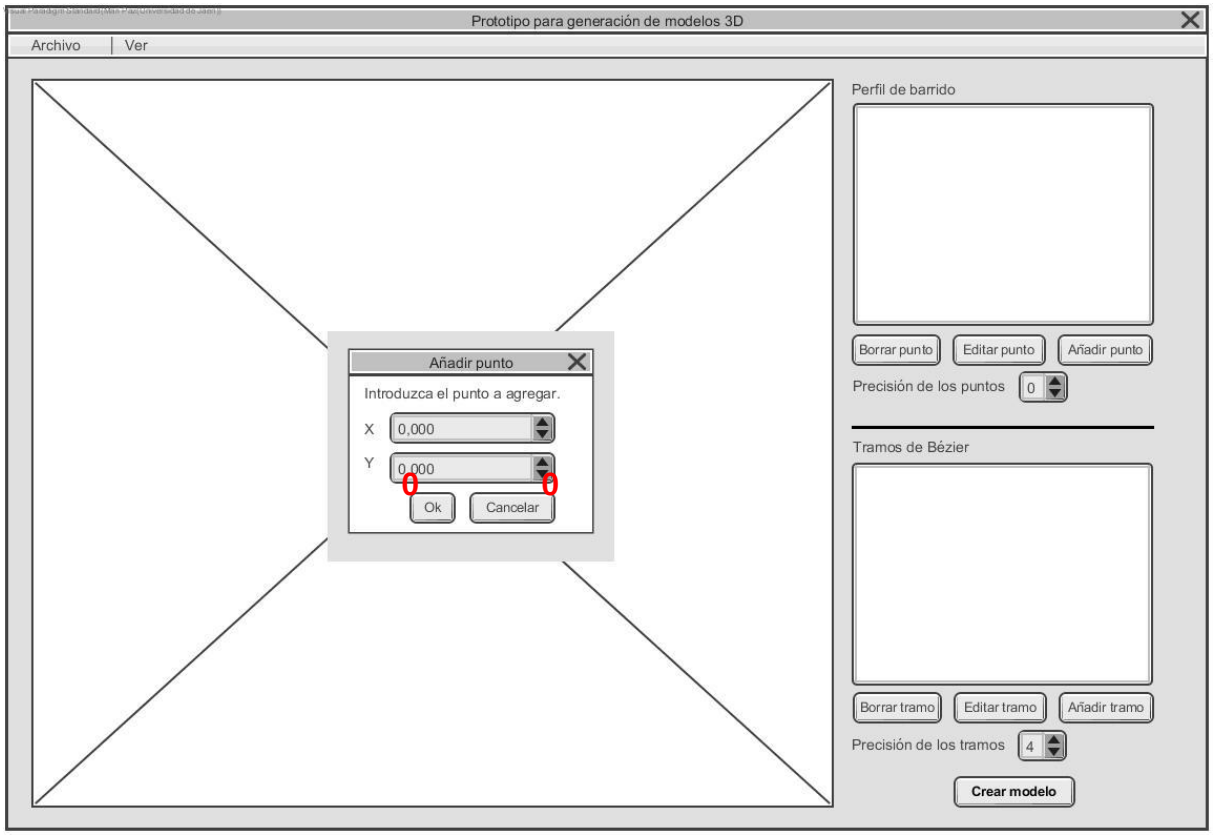
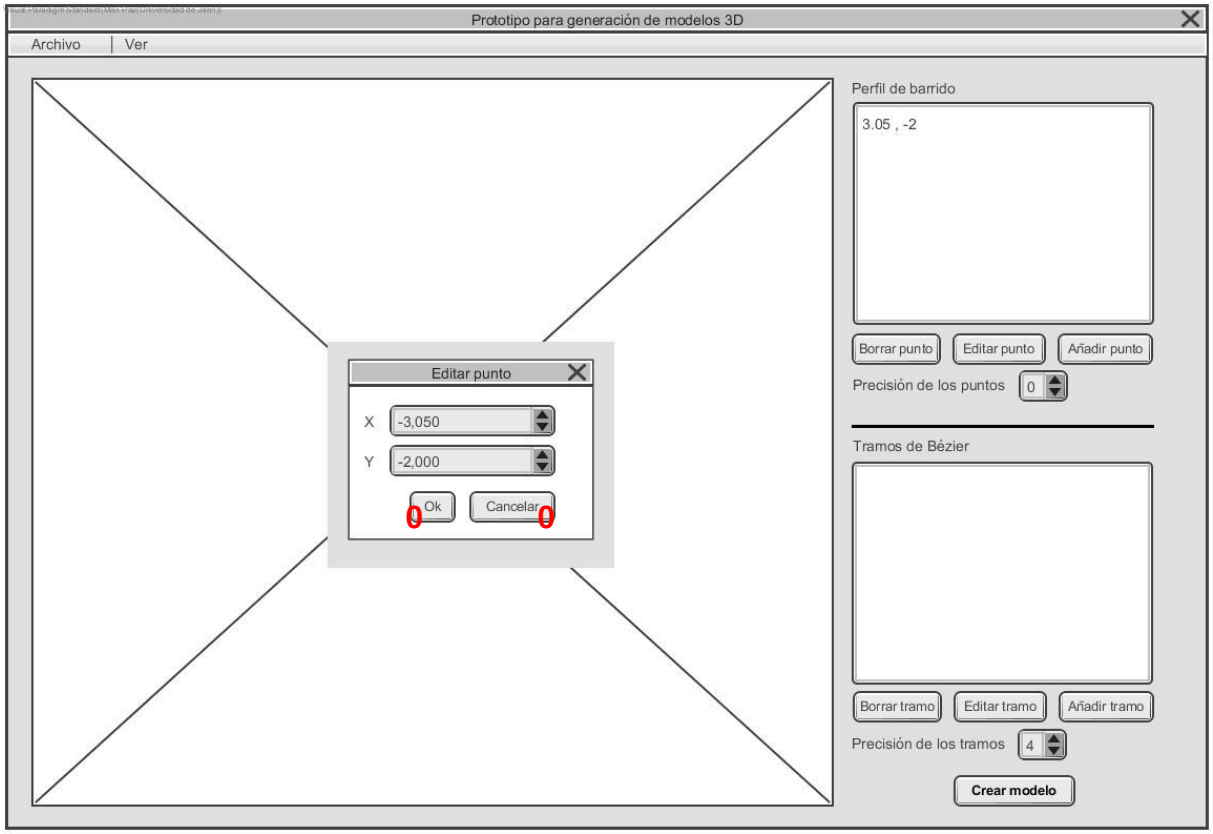
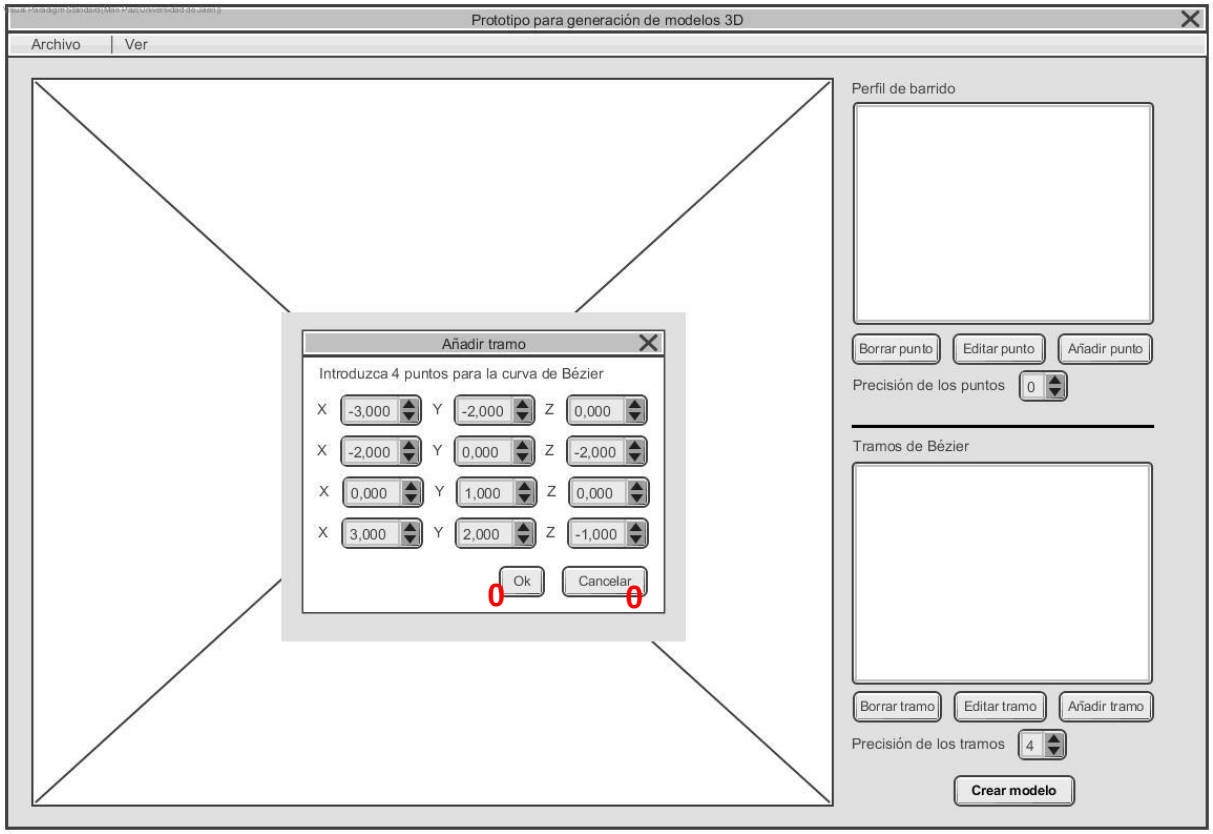


Ilustración 2.13. Storyboard pantalla 1



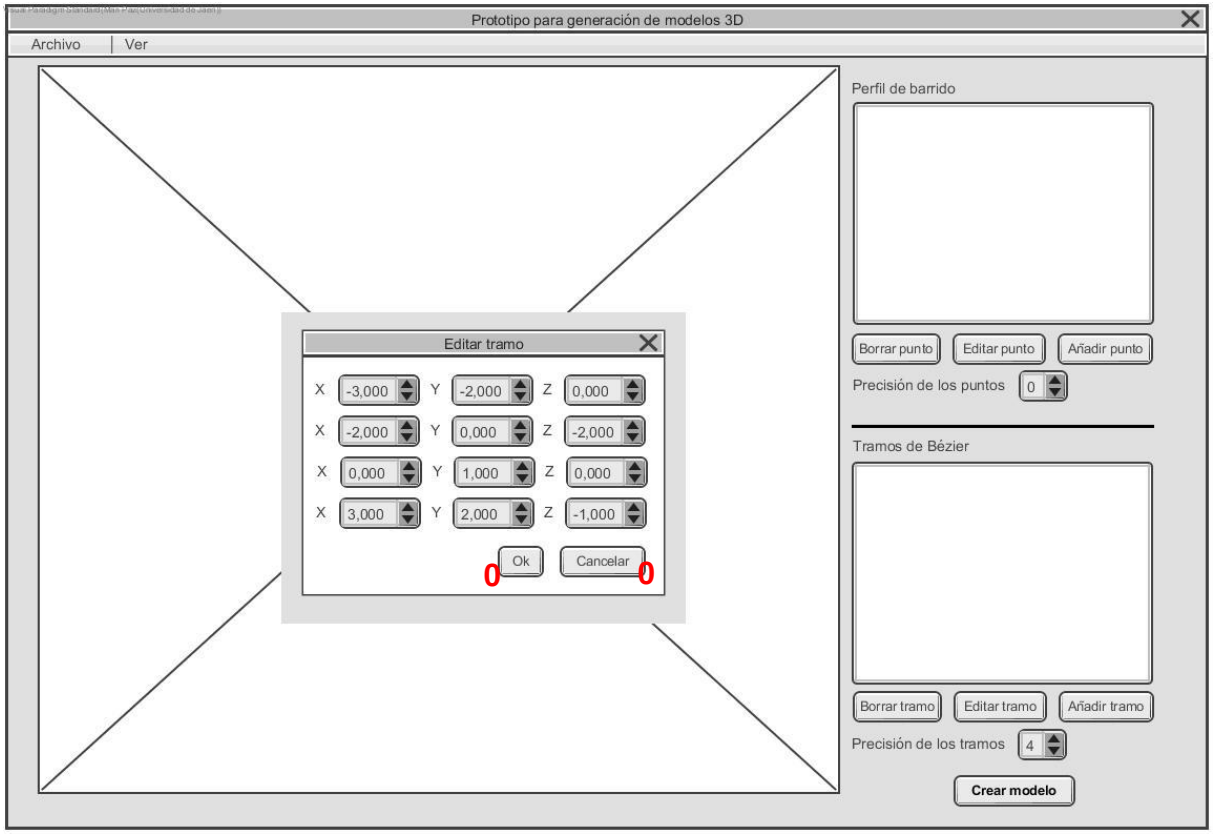
2

Ilustración 2.14. Storyboard pantalla 2



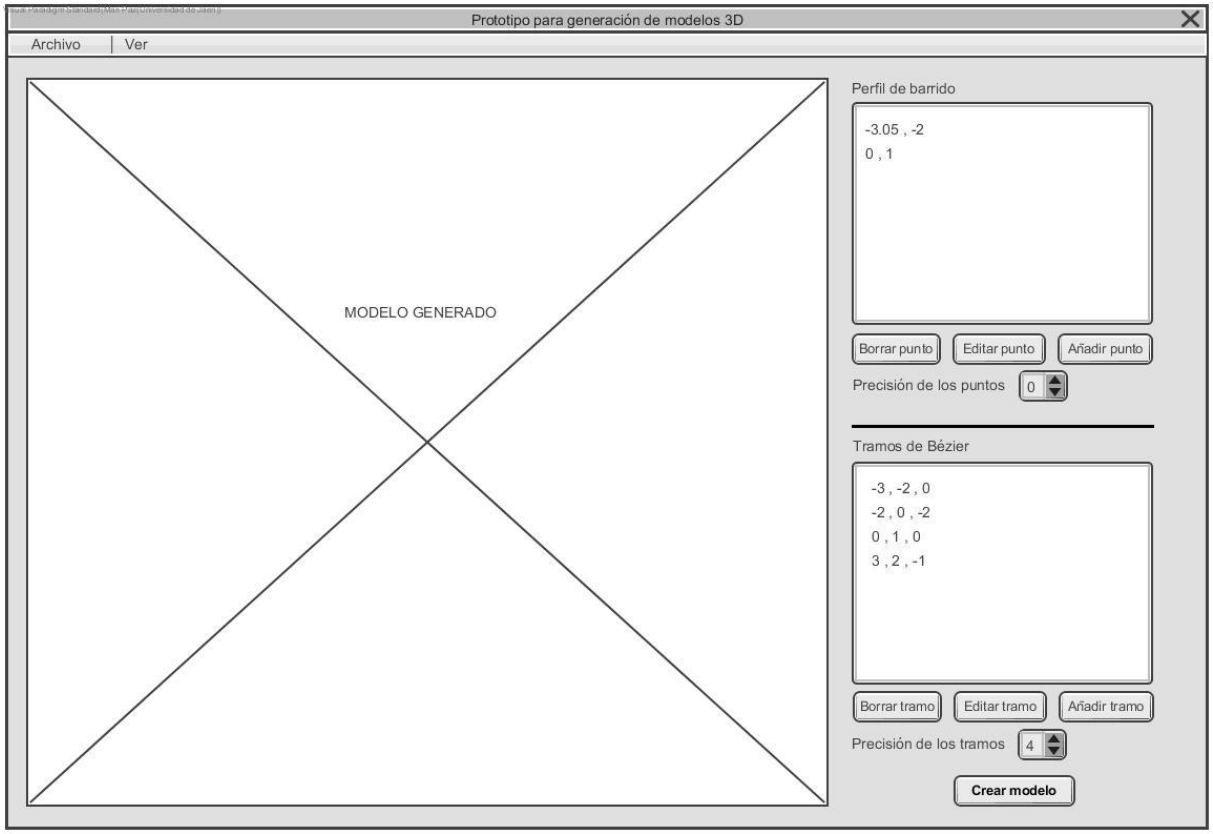
3

Ilustración 2.15. Storyboard pantalla 3



4

Ilustración 2.16. Storyboard pantalla 4



5

Ilustración 2.17. Storyboard pantalla 5

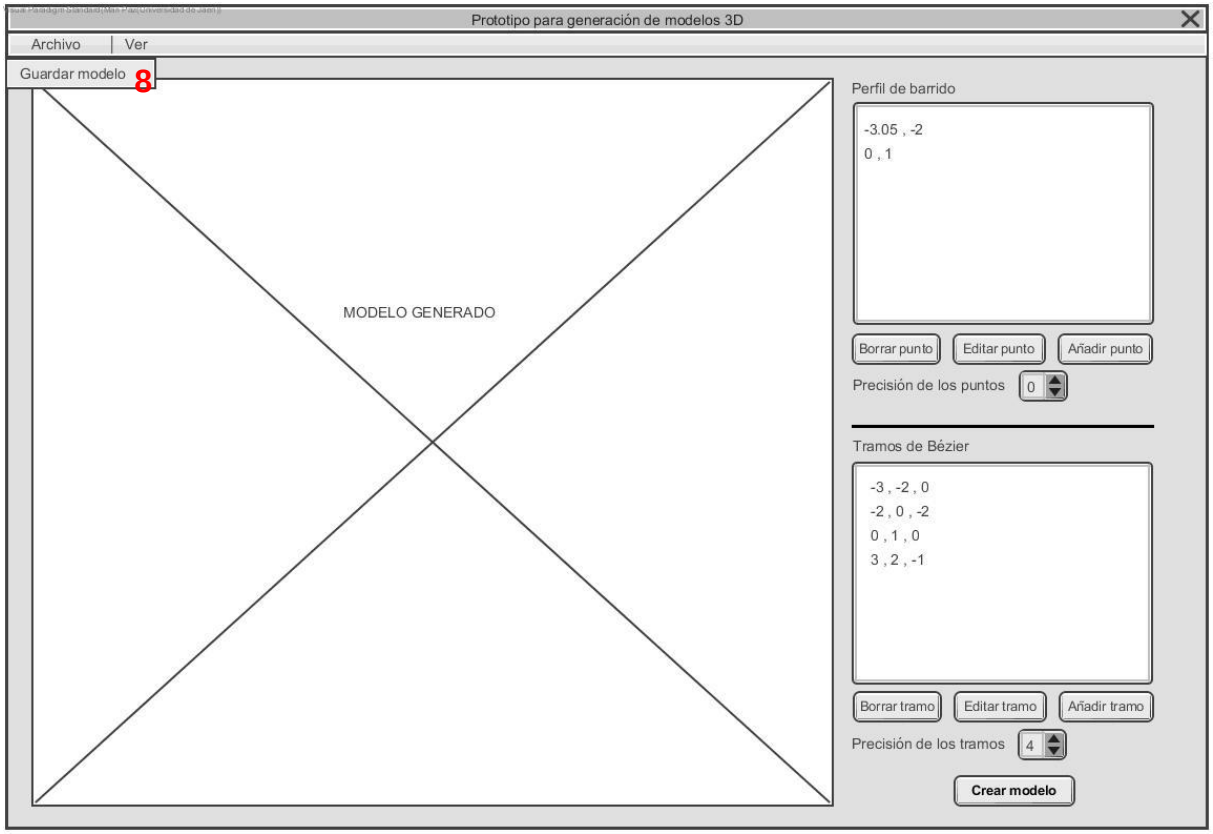


Ilustración 2.18. Storyboard pantalla 6

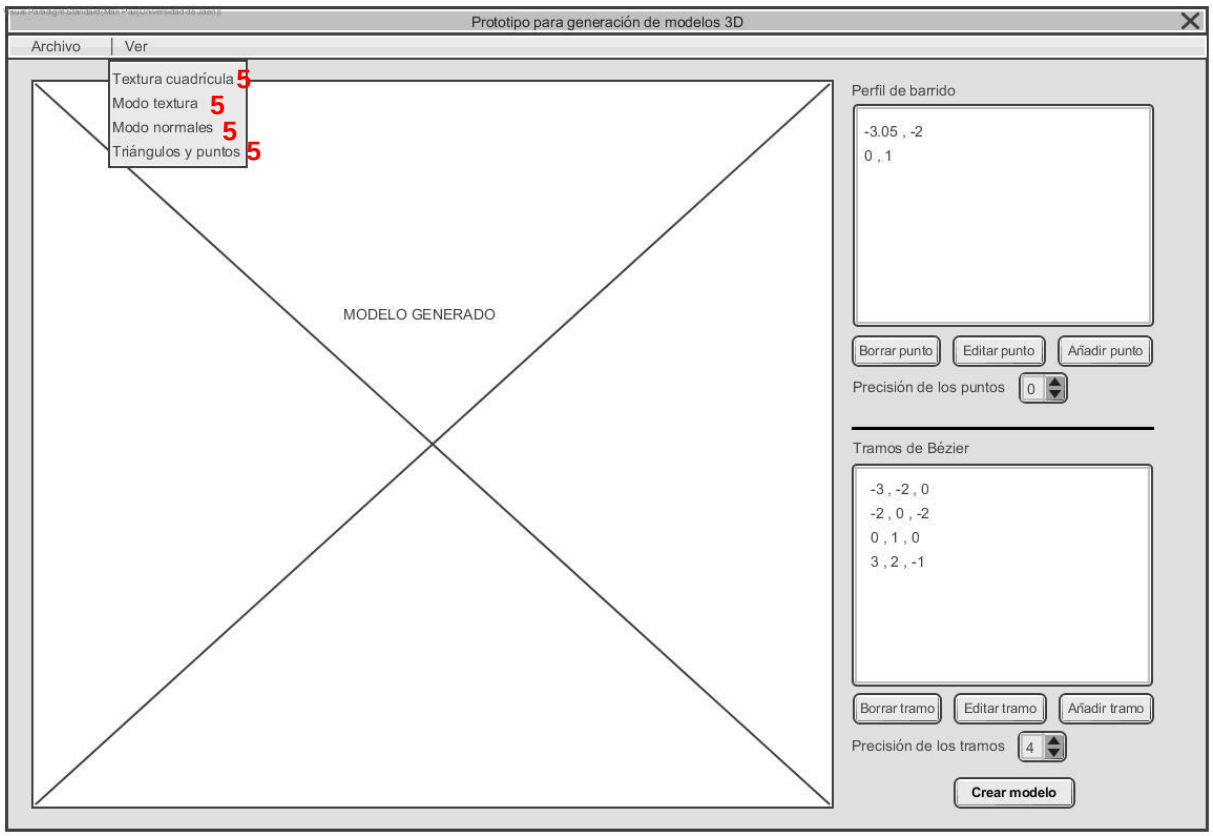


Ilustración 2.19. Storyboard pantalla 7

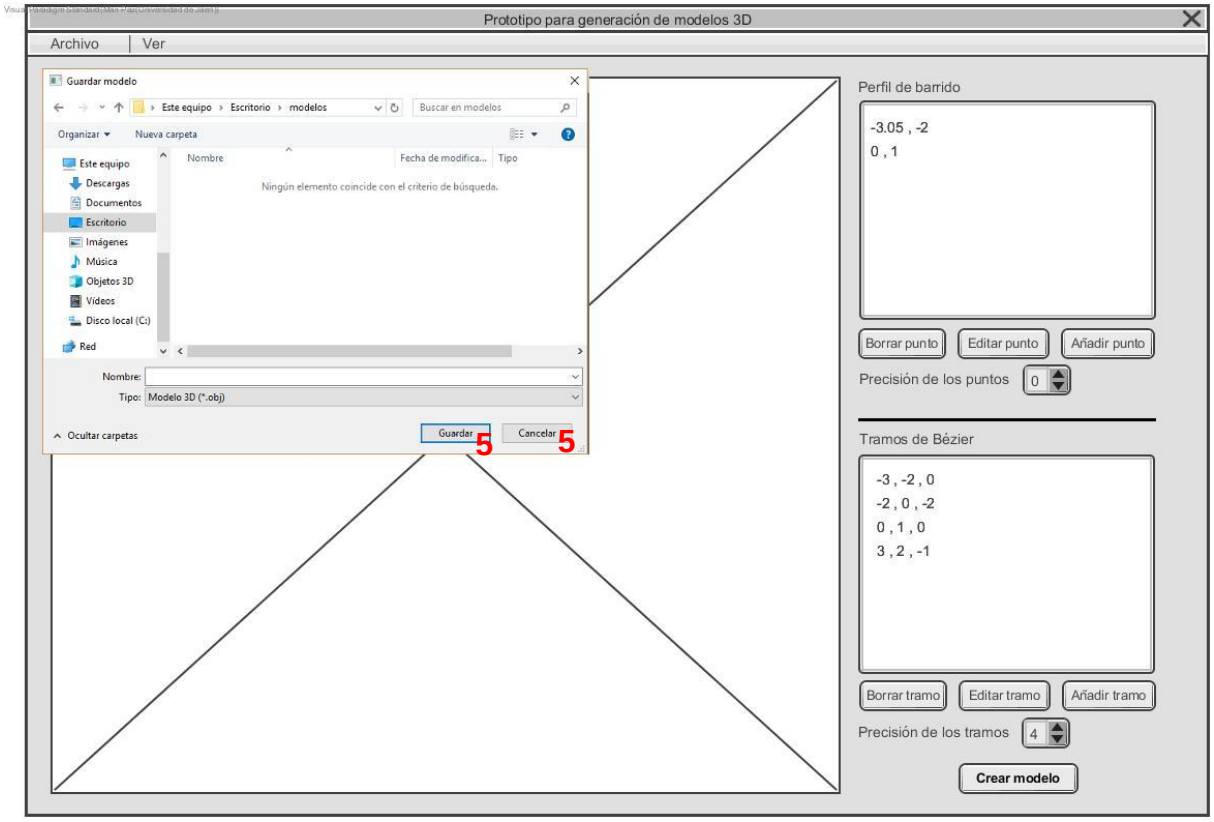


Ilustración 2.20. Storyboard pantalla 8

2.3 - Implementación

Como se ha mencionado en apartados anteriores, la aplicación parte de una primera aplicación desarrollada en las prácticas de una asignatura. Sin embargo, la primera aplicación no disponía de una interfaz gráfica, elemento indispensable en la nueva aplicación. Por este motivo, el primer paso en la implementación fue la búsqueda de una biblioteca que permitiera la creación de la interfaz gráfica además del soporte para OpenGL. Después de una búsqueda de la biblioteca que mejor se adaptara a los requisitos, se concluyó que la mejor opción era usar Qt. Esto supuso el hecho de tener que cambiar parte del proyecto existente para adaptarlo a Qt.

Este cambio a Qt trajo consigo pequeños cambios en el código como pueden ser los nombres de las estructuras de datos y de los tipos de datos, así como el cambio de algunos métodos que provee la biblioteca. Sin embargo, a la hora de gestionar la ventana en la que se va a dibujar la geometría el cambio ha sido mayor, ya que, para poder incluir esta ventana en la ventana principal de la aplicación, debe heredar de QWidget (en Qt los elementos de la interfaz gráfica son considerados widgets). De esta manera, para encapsular el comportamiento de la ventana de OpenGL, se ha

creado una clase que hereda de QOpenGLWidget e implementa los métodos relacionados con el dibujado de la escena.

El siguiente paso en la implementación es la creación de la interfaz gráfica en la que se presentará toda la información al usuario. Para ello se crea una clase que hereda de QMainWindow. Con la ayuda del apartado Design de Qt Creator se han añadido los elementos de forma gráfica en la interfaz para facilitar esta tarea. Esta clase tiene varios atributos para almacenar la información que el usuario va a introducir. También pasa esa información a la clase Renderer, que se detallará más adelante, para que esta procese los datos.

Antes de pasar a las clases que generan los modelos se va a justificar las decisiones que se han tomado sobre los mismos.

En primer lugar, los modelos se van a definir mediante dos tipos de datos. Por una parte, tenemos puntos pertenecientes a un plano 2D que se van a denominar perfil o perfil de barrido. El perfil es lo que va a aportar gran parte de forma al modelo. A la hora de construir el modelo se van a tomar los puntos del perfil según su orden, teniendo en cuenta la rotación CCW. Por este motivo hay que tener especial cuidado a la hora de diseñar e introducir los puntos.

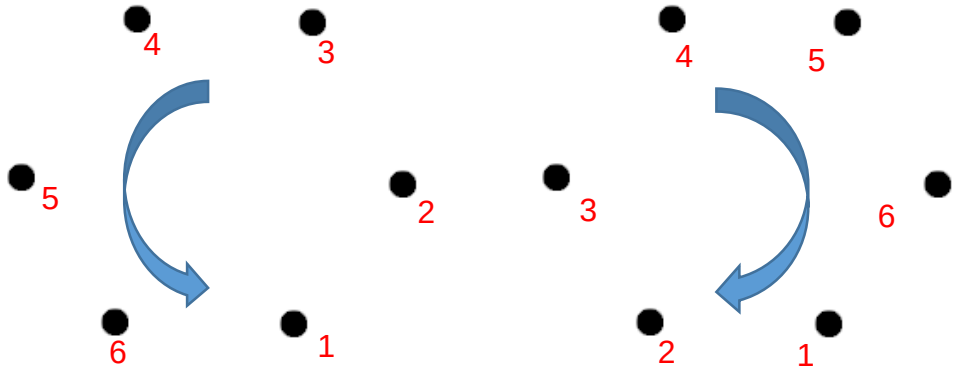


Ilustración 2.21. Ejemplo de puntos en sentido anti horario (izquierda) y horario (derecha)

Por otra parte, para formar el modelo necesitamos una trayectoria sobre la que extruir o barrer los puntos del perfil. Para ello se ha decidido emplear curvas de Bézier, ya que presentan numerosas ventajas nombradas en apartados anteriores frente a otros tipos de curva.

A continuación, se procede a detallar las clases que se han utilizado para construir los modelos. Son tres: ruta, tramo y superficie.

En primer lugar, tenemos a la clase ruta, que representa a la trayectoria completa que van a seguir los puntos del perfil para formar el modelo. Para poder aportar mayor complejidad a esta trayectoria se ha decidido partirla en tramos más pequeños y más sencillos de definir y procesar. Por este motivo, uno de los atributos más relevantes de esta clase es un array con los tramos que componen la ruta.

Por otra parte, los podemos distinguir los métodos que implementa esta clase como:

- Métodos “drawAs”: se utilizan a la hora del dibujado de cada tramo de la ruta. Hay varios, dependiendo de si se va a dibujar triángulos, puntos, líneas…
- Método para guardar el modelo: se utiliza cuando el usuario quiere salvar los datos a un fichero. En este procedimiento se obtienen los datos de la geometría y topología de cada tramo y se escriben en el fichero.
- Método para obtener las coordenadas de textura de cada tramo: se utiliza para obtener las coordenadas de textura relativas al modelo total para cada tramo. Así, se evita que la textura genere sensaciones extrañas cuando los tramos tienen longitudes dispares, o, en otras palabras, que la textura se vez en unos tramos más expandida que en otros.

Cabe destacar que, a la hora de definir los puntos que van a dar forma a la curva que representa a la ruta, el usuario se va a encontrar con el hecho de que la primera vez debe introducir cuatro puntos para definir la curva y en los demás casos sólo tiene que introducir dos puntos. Esto se debe a la continuidad paramétrica de orden uno de la que se ha decidido dotar al modelo. De esta forma, el primer punto del segundo tramo de la curva corresponde con el último punto del primer tramo y el segundo punto del segundo tramo se construye mediante el punto simétrico del tercer punto con respecto al cuarto, ambos del primer tramo. Así, los dos puntos que introduce el usuario corresponden con el tercer y cuarto punto del segundo tramo.

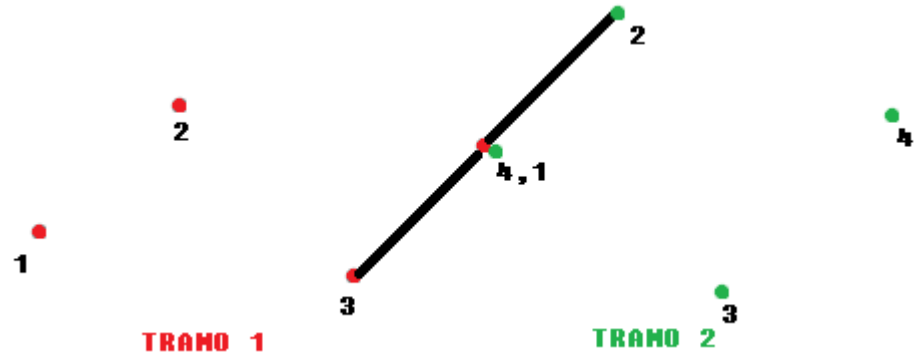


Ilustración 2.22. Ejemplo de generación de los puntos del segundo tramo

Cuando la ruta se crea con todos los puntos del perfil y de la curva, esta misma se encarga de los cálculos de los puntos 1º y 2º de los tramos (en caso de que sea necesario) y va formando los tramos con los cuatro puntos correspondientes. Así se asegura el cumplimiento de la continuidad C^1 previamente descrita.

Pasamos ahora a la clase tramo, que representa una curva de Bézier definida por cuatro puntos. Esta clase tiene diferentes atributos referidos a la geometría de la propia curva y su dibujado. Sirve para visualizar la curva y los cuatro puntos que la han generado. Para ello contiene una serie de estructuras de datos que almacenan los puntos generadores, los puntos calculados dependiendo de la precisión que el usuario haya indicado para la curva, y otras para almacenar los índices que determinan las relaciones entre los puntos, o sea, la topología de la curva. Para el dibujado de la curva es necesario un VAO, por lo que se introducen dos como atributos. Uno hace referencia a los puntos de la curva y otro a los puntos generadores de la curva.

Los métodos más relevantes de esta clase son:

- Bézier: se encarga de calcular los puntos de la curva mediante un parámetro t que va desde 0 hasta la precisión que indique el usuario. Para ello hace uso de las funciones mezcla de Bézier.
- Un procedimiento para rellenar los VBO de cada VAO: para especificar el tipo de información de la geometría que contiene cada VBO e introducirla dentro del mismo.
- Un procedimiento para rellenar los IBO de cada VAO: en este caso se introduce información de tipo topológica, o lo que es lo mismo, los

índices que van a indicar cómo se dibuja la geometría. En este caso sólo se van a rellenar los IBO referentes a líneas y puntos.

- Métodos drawAs…: sirven para dibujar la información que contiene el VAO según el modo que se haya seleccionado (puntos o líneas). Además, se pasan algunos datos a los shaders de dibujo, como la matriz de modelado, visión y proyección.

Por último, cada tramo dispone de un atributo de la clase superficiePerfil, que representa la porción de modelo que le corresponde a cada tramo. De esta forma, el modelo total viene determinado por el conjunto de modelos de cada uno de los tramos en los que se descompone la curva.

La clase superficiePerfil se encarga de generar la geometría del modelo que representa a un tramo barriendo los puntos del perfil por la trayectoria que define la curva de los puntos del tramo. Para ello, cuenta con los atributos:

- Estructuras de datos para almacenar la información de los puntos del perfil, los puntos generadores de la curva, los puntos del modelo calculados y sus coordenadas de textura y normales correspondientes, además de los índices que indican la topología.
- Para dibujar el modelo que se va a construir, es necesario un VAO, además de otro que se va a crear para visualizar otros puntos relevantes.

Por otra parte, para generar el comportamiento de esta clase se ha hecho:

- En primer lugar, para obtener un perfil de barrido con curvas más suaves se ha hecho uso de la clase ya existente subdivisionProfile. Esta clase, dados unos puntos que forman un perfil, genera un conjunto de puntos cuya geometría es más suave (no presenta vértices tan abruptos) haciendo uso de una media ponderada. Así, para obtener los punto intermedios (todos menos el primero y el último), habría que realizar las siguientes operaciones:

$$\frac{p_{i-1} + p_i}{2}$$

$$\frac{3p_i}{4} + \frac{p_{i-1}}{8} + \frac{p_{i+1}}{8}$$

De esta manera, los puntos pasarían por esta transformación:

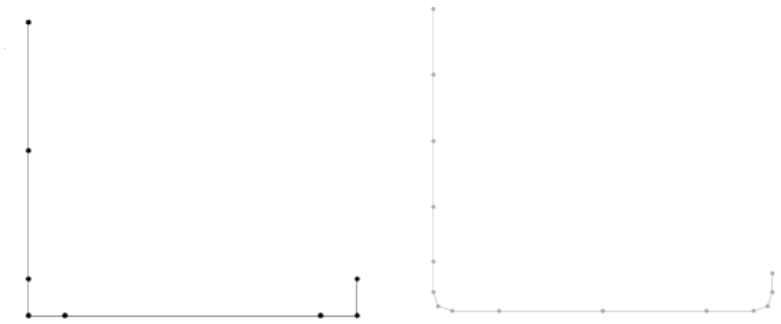


Ilustración 2.23. Ejemplo de subdivisión de perfil. Extraída de los apuntes de PAG

- Un método para generar la superficie del modelo. Para ello, el procedimiento se basa en evaluar la curva en un punto y con ayuda de la primera derivada de las funciones mezcla de Bézier y otra curva de Bézier, que indica la dirección de referencia, obtener una matriz de transformación que lleve los puntos del perfil alrededor del primer punto evaluado de la curva. Así, los puntos que se obtienen van cogiendo la forma de la curva que forma la ruta. De esta forma, la matriz de transformación para cada punto de la curva de Bézier de cada tramo quedaría así:

$$\begin{matrix} u_x & v_x & n_x & p_x \\ u_y & v_y & n_y & p_y \\ u_z & v_z & n_z & p_z \\ 0 & 0 & 0 & 1 \end{matrix}$$

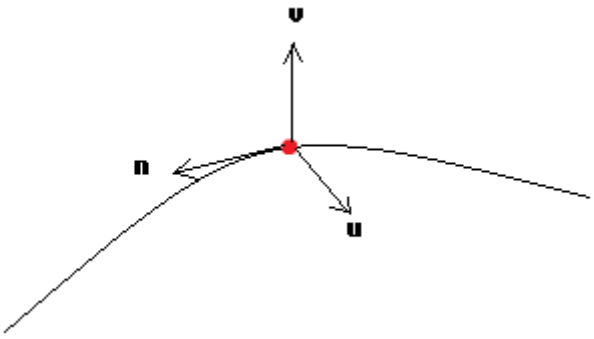


Ilustración 2.24. Ejemplo de vectores que representan el sistema de coordenadas del punto señalado

Donde:

- P(p_x, p_y, p_z) viene determinado por cada uno de los puntos evaluados por las funciones mezcla de Bézier,
- **n**(n_x, n_y, n_z) se calcula como el vector en sentido contrario al obtenido al evaluar las funciones mezcla derivadas,
- **u**(u_x, u_y, u_z)[2] se obtiene, en primer lugar, evaluando la segunda curva (cada punto representa un vector dirección para cada punto evaluado en la primera curva) y calculando un vector que une P con su punto correspondiente de la segunda curva para obtener el vector dirección. Después, se realiza el producto vectorial entre ese vector y el vector **n** ya calculado.
- **v**(v_x, v_y, v_z) se calcula como el producto vectorial entre **n** y **u**.

Una vez calculada la matriz, se multiplican cada uno de los puntos del perfil y sus normales correspondientes por la matriz. Así se llevan al "espacio de coordenadas" del punto de la curva que se está evaluando.

- Un método para calcular las normales de los puntos del perfil. Para ello se hacen los siguientes cálculos:

1. Se calcula el vector que forma un punto del perfil con su anterior y su siguiente.

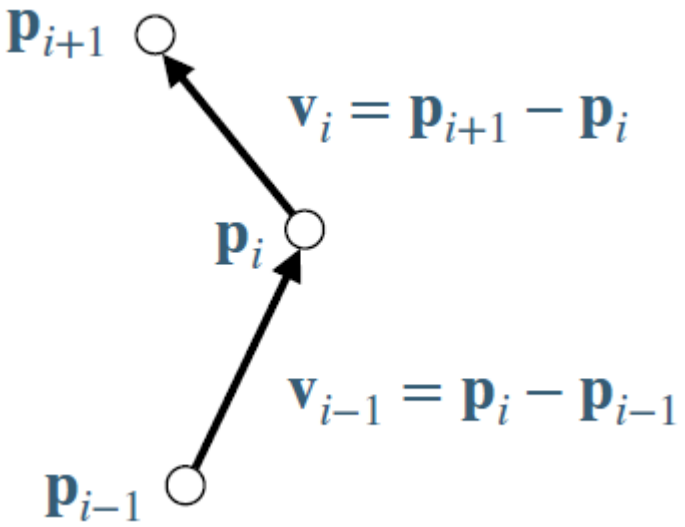


Ilustración 2.25. Ejemplo I de cálculo de normales de un perfil. Extraída de los apuntes de PAG

[2] En este caso, para obtener la segunda curva, se han calculado cuatro puntos generadores partiendo de los cuatro que generaban la curva inicial sumándoles un épsilon.

2. Una vez tenemos los dos vectores calculados, se rotan 90º.

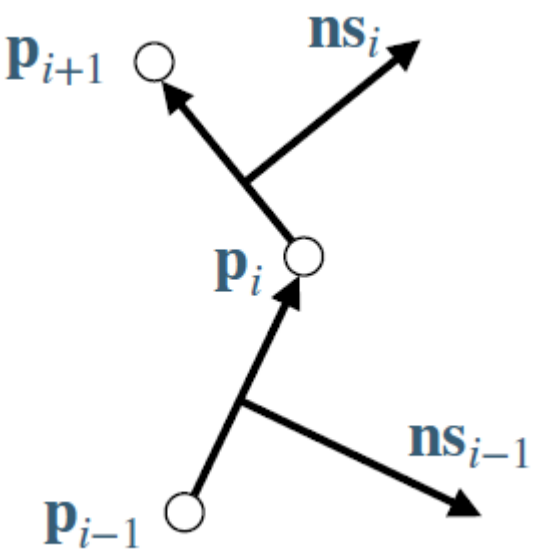


Ilustración 2.26. Ejemplo II de cálculo de normales de un perfil. Extraída de los apuntes de PAG

3. Con esto tendríamos las normales a los vectores iniciales. Si sumamos estas normales y dividimos entre 2, se obtiene la normal para el punto P_i.

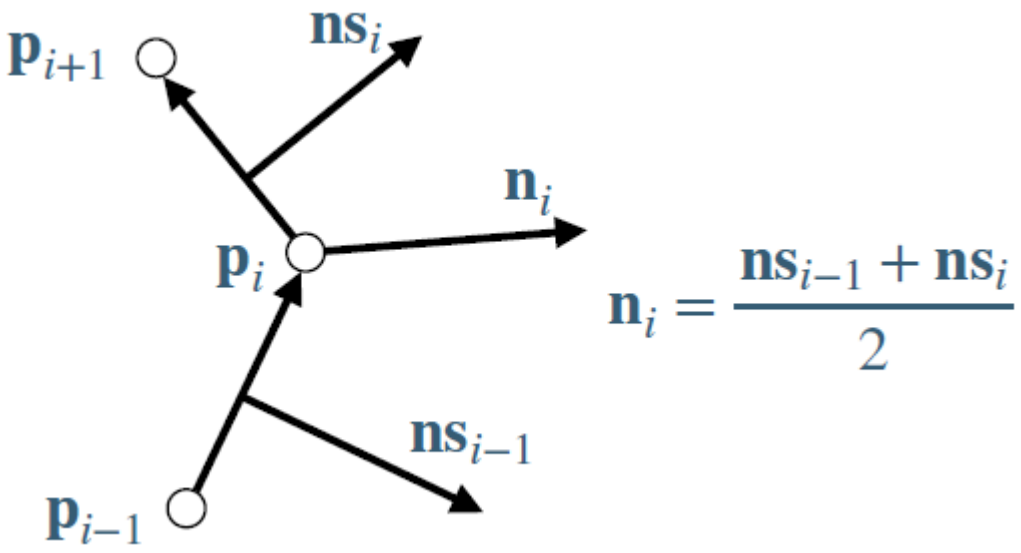


Ilustración 2.27. Ejemplo III de cálculo de normales de un perfil. Extraída de los apuntes de PAG

4. Para el primer y el último punto la normal sería la calculada en su segmento correspondiente.

- Otra operación relevante en esta clase es la que calcula los índices de los triángulos que forman el modelo. En este punto hay que tener en cuenta que los vértices del triángulo deben estar en sentido contrario a las agujas del reloj para que las normales sean correctas. Para ello se ha implementado un algoritmo que, dados dos puntos consecutivos del perfil, obtiene los índices de los vértices calculados correspondientes a ambos puntos tal que formen triángulos con la rotación CCW para después ser dibujados mediante la primitiva GL_TRIANGLE_STRIP.

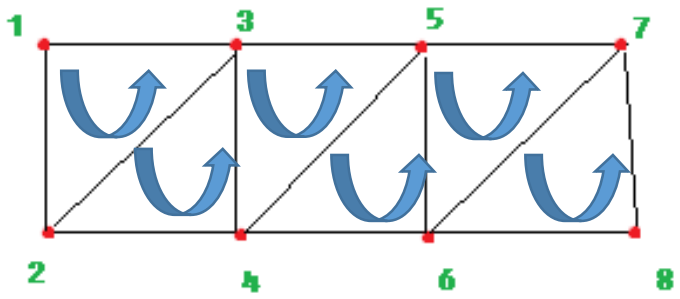


Ilustración 2.28. Ejemplo de tira de triángulos (triangle strip)

- También es necesario el cálculo de las coordenadas de textura para cada uno de los vértices calculados. Como un modelo puede tener más de un tramo, hay que calcularlas con respecto al modelo en total, de forma que un tramo abarque las coordenadas que le corresponden del total de la textura. Para ello, cada uno de los tramos calcula su longitud y la ruta a la que pertenecen calcula la proporción entre la longitud de cada tramo y la total. Como consecuencia, cada tramo calcula sus coordenadas con respecto a un punto inicial y un punto final. De esta forma se calcula la coordenada u de la textura. La coordenada v se calcula como la longitud acumulada de un punto del perfil entre la longitud del perfil completo.

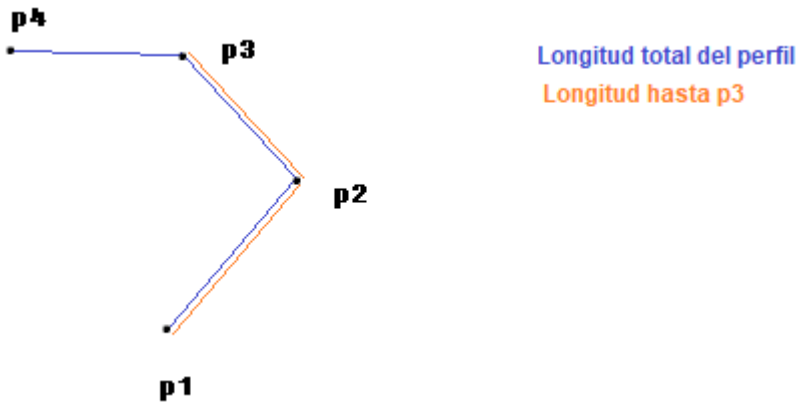


Ilustración 2.29. Ejemplo de cálculo de longitud del perfil

Ilustración 2.30. Textura de cuadrícula. Extraída de los apuntes de PAG

- A la hora del dibujado del modelo necesitamos introducir la geometría y la topología en un VAO. Por una parte, tenemos un procedimiento que especifica e introduce la geometría en los VBO. Por otra parte, la topología se indica en un método que introduce los índices calculados en los IBO.

- Por último, los métodos drawAs… que ya se han mencionado en otras clases y tienen un comportamiento similar. En ellos se dibuja la información contenida en el VAO previamente creado.

La clase renderer es una de las clases principales de la aplicación. Esta clase se ha implementado como un singleton. Implementa el método al que se llama en el dibujado de la ventana de OpenGL y gestiona gran parte del comportamiento de los eventos que se producen en la misma. Sus atributos principales son los shaders, la ruta que se va a dibujar una vez creada y una cámara que nos va a permitir visualizar y movernos alrededor del modelo.

Sus métodos principales son:

- prepareOpenGL: se encarga de preparar la escena para su visualización. Por ello, debe ser llamado antes de que comience el ciclo de eventos. En él se cargan y crean los shader programs que luego se usarán en el dibujado. Además, se activan órdenes de OpenGL referentes al dibujado posterior.
- refreshCallback: este método se llama cuando se quiere redibujar la escena. Para ello, se limpian los buffers de color y profundidad.
- redimensionarAreaDibujo: se encarga de cambiar las dimensiones del viewport y de pasar las nuevas dimensiones a la cámara para cambiar su relación de aspecto. Así, cuando la ventana principal cambie de tamaño, no se notará ningún artefacto en la ventana de OpenGL.
- crearRuta: gracias a este método, la ventana principal puede mandar al renderer la información que ha introducido el usuario mediante la interfaz para crear el modelo. En ese momento se crea la nueva ruta que va a dar paso al modelo. Esta ruta se va a almacenar en un atributo para que después se dibujen sus tramos y superficies.
- El método dibujar es el encargado de seleccionar el shader apropiado y, con las matrices de visión y proyección de la cámara, dar la orden a la ruta para que se dibuje con los parámetros seleccionados. En esta función se distinguen diferentes modos a la hora de pintar, que serán los que luego pueda seleccionar el usuario desde la interfaz.

En primer lugar, se encuentra el modo nube de puntos, que va a dibujar el modelo con todos los vértices que lo componen, además de los puntos que han generado ese modelo. En este modo también se puede visualizar la línea que representa a la curva de Bézier que ha dado forma a cada tramo.

En segundo lugar, se ha implementado el modo malla de triángulos, en el que se visualiza la superficie del modelo con una textura de cuadrícula. Así, se da una mayor sensación de ubicación espacial del modelo.

A continuación está el modo normales. En él se visualiza la superficie del modelo también, pero esta vez los colores codifican las normales de cada fragmento del modelo. Las normales se representan con un vector de tres componentes, que pueden ocupar las tres componentes rgb del color. Sin embargo, los canales rgb comprenden un intervalo de valores entre 0 y 1 (ambos incluidos), y las normales pueden comprender valores entre -1 y 1. Por tanto, se han transformado las normales en el shader para adaptarlas al rango de los colores.

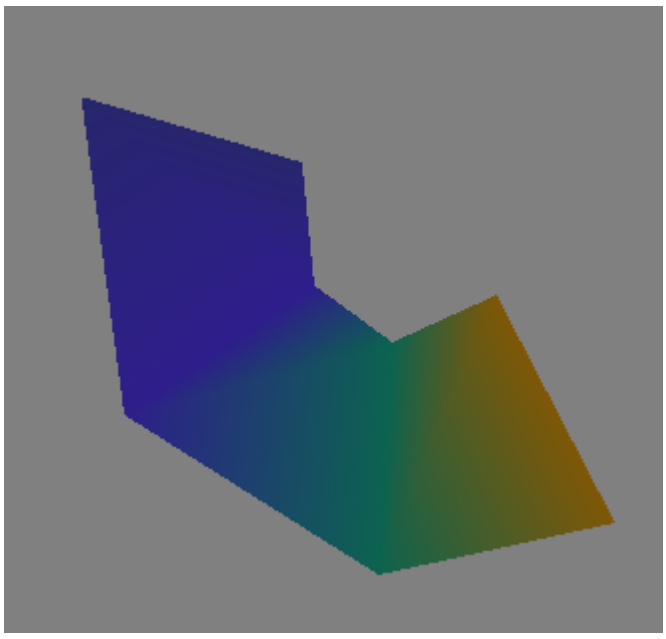


Ilustración 2.31. Ejemplo de modelo en modo normales

$$color = ((normal * 0.5) + 0.5{,}1.0)$$

El último modo que incorpora es el modo textura. De esta manera se pueden visualizar las coordenadas de textura codificadas como color. En este caso, las coordenadas u y v toman valores entre 0 y 1, por lo que no es necesaria ninguna transformación. Simplemente se ha tomado como color rojo la coordenada u y como color verde la coordenada v. El color azul se ha fijado a 1 ya que sólo hay dos coordenadas de textura.

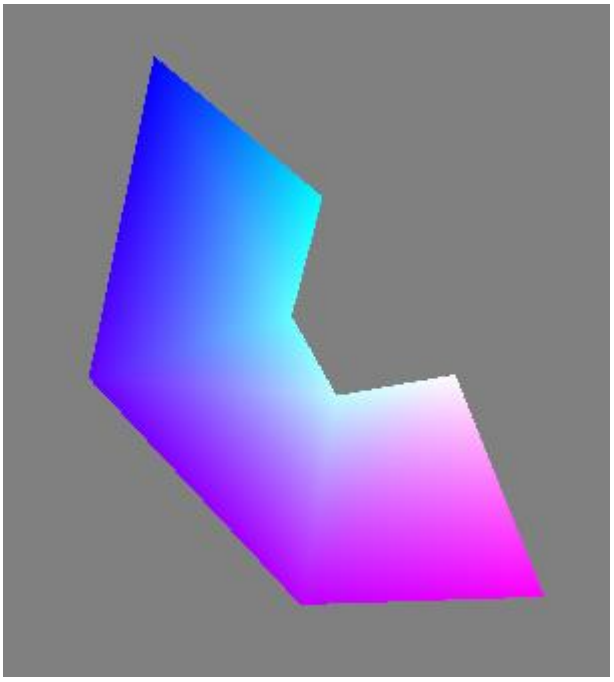


Ilustración 2.32. Ejemplo de modelo en modo coordenadas de textura

2.4 - Pruebas finales

2.4.1 - Pruebas de verificación del sistema

Una vez la aplicación se ha desarrollado, la aplicación permite:

- Añadir puntos que van a pertenecer al perfil.
- Modificar y borrar los puntos del perfil ya añadidos que se deseen.
- Introducir el grado de suavidad que se quiere para el perfil.
- Añadir puntos en tramos que van a formar la curva de la ruta.

- Modificar los puntos de un tramo.
- Borrar tramos.
- Especificar el grado de subdivisiones que va a tener la curva.
- Crear modelos a partir de los puntos y tramos introducidos.
- Poder visualizar el modelo creado en diferentes modos que muestran sus características.
- Poder almacenar los datos del modelo en un archivo.

En conclusión, podemos afirmar que se han cumplido los objetivos funcionales de la aplicación.

2.4.2 - Pruebas de validación del sistema

El sistema cumple el requisito fundamental con el que fue ideada, la usabilidad. La aplicación es sencilla e intuitiva, y muestra la información indispensable en cada momento. Por esta razón se puede decir que un usuario que no ha tenido ningún contacto con alguna aplicación de este tipo, no tendrá problemas para aprender a utilizarla en un corto espacio de tiempo.

Por otra parte, la aplicación ofrece tiempos de respuesta imperceptibles por el usuario, siendo así inmediata la visualización de los cambios.

2.5 - Resultados obtenidos

Algunos modelos obtenidos como resultado de la aplicación son:

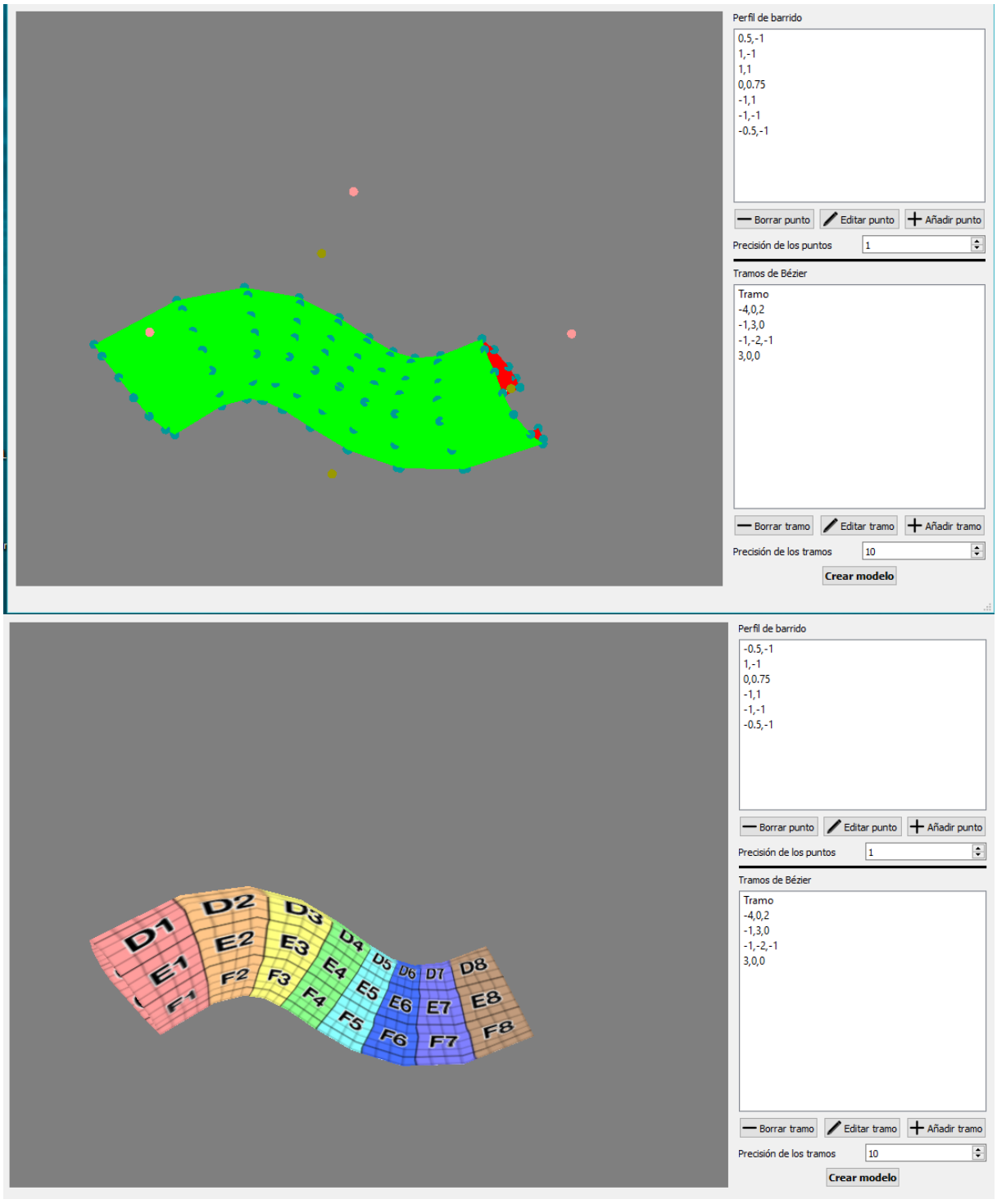
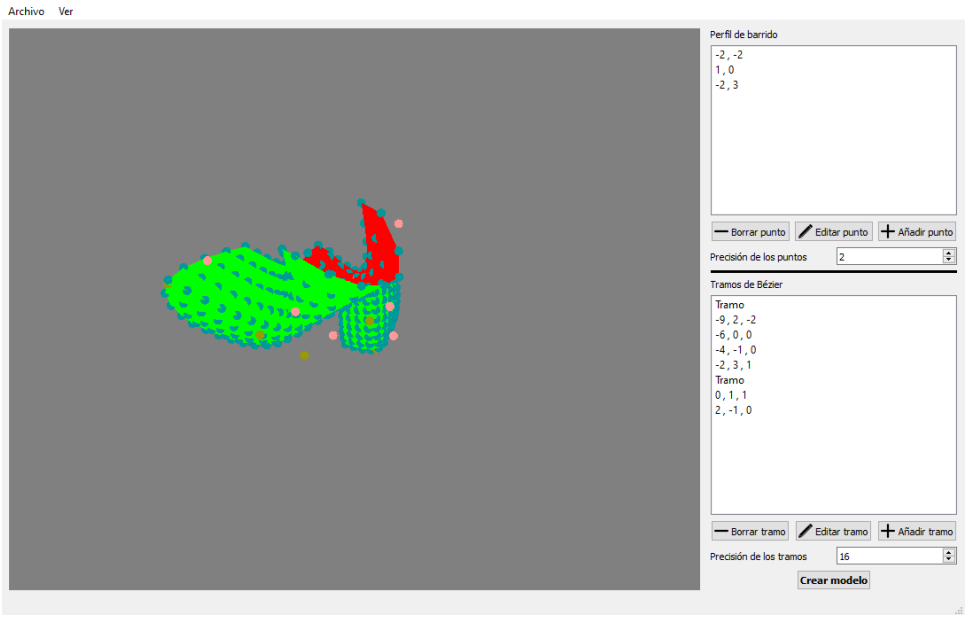


Ilustración 2.32. Ejemplo de modelo generado



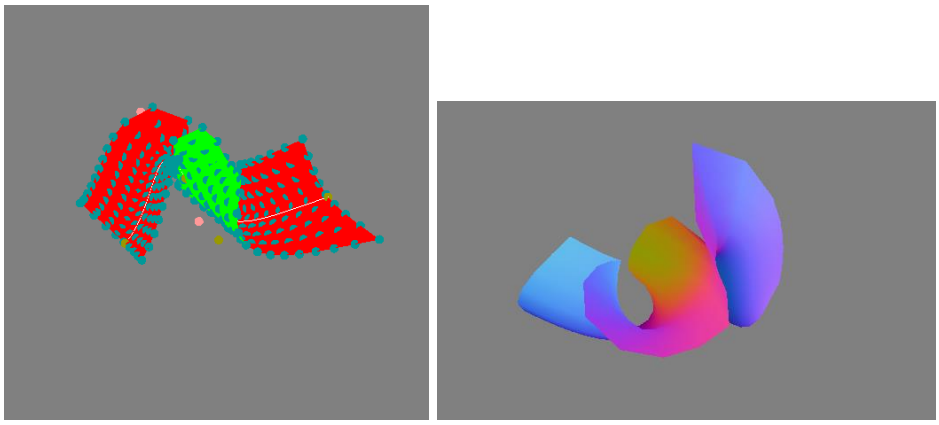


Ilustración 2.33. Ejemplo de modelo generado

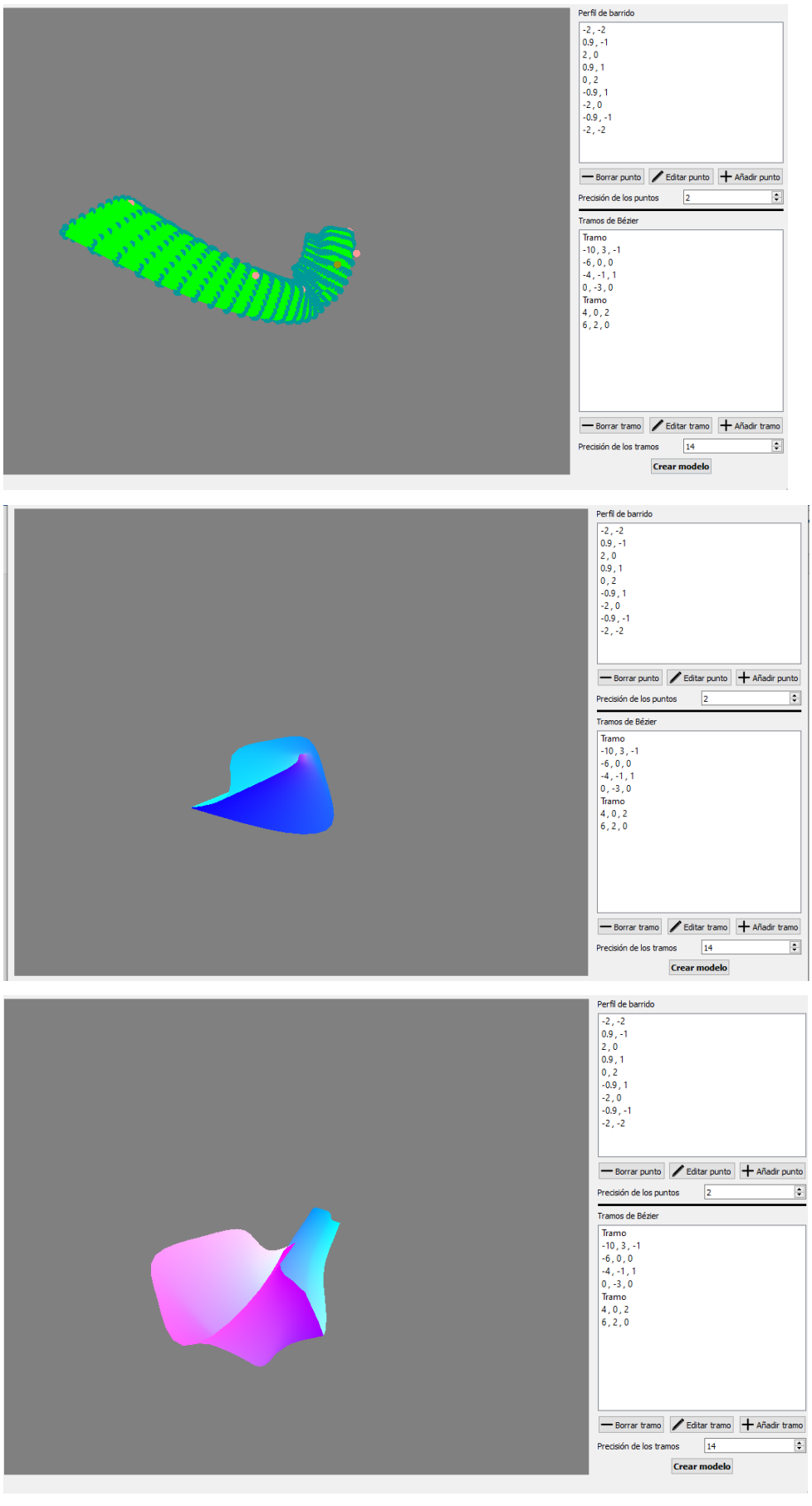


Ilustración 2.34. Ejemplo de modelo generado

3 - CONCLUSIONES Y TRABAJOS FUTUROS

En relación a los apartados anteriores, se puede afirmar que la aplicación ha logrado satisfacer los objetivos con los que fue concebida. De esta manera se ofrece una alternativa sencilla que aporta una ayuda en la didáctica.

En un futuro se podrían añadir nuevas funcionalidades al sistema. Por ejemplo, la posibilidad de cargar nuevos shaders y texturas para aplicar a los modelos. Además, se podrían realizar pruebas observando cómo interactúa el usuario con la aplicación y dónde comete errores para así solventar las posibles dudas que se puedan generar.

4 - APÉNDICES

4.1 - Instalación y configuración del sistema

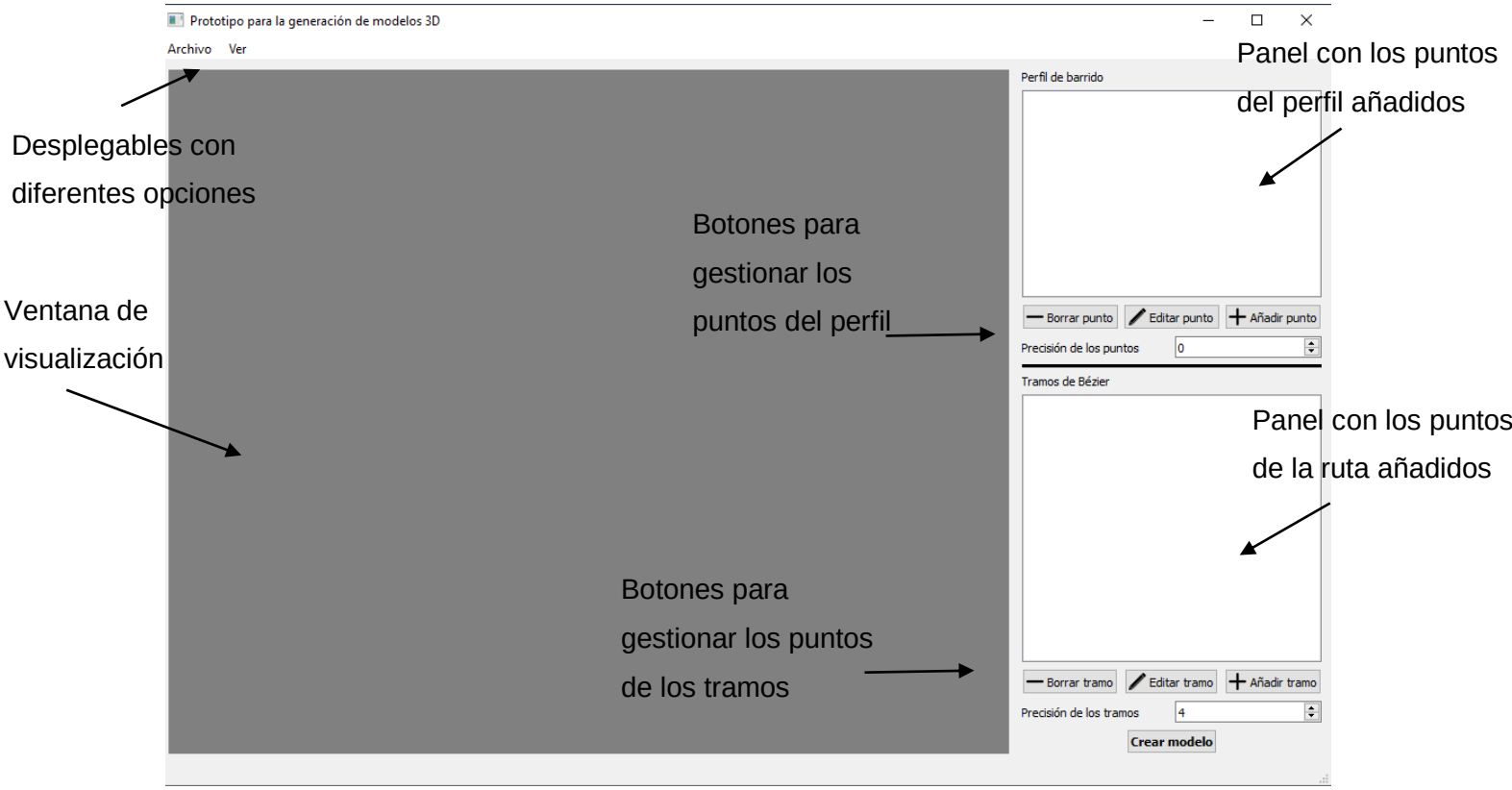
Para hacer uso de la aplicación se deben seguir los siguientes pasos:

1. Descomprimir la carpeta y acceder a la carpeta llamada *build-tfg-Desktop_Qt_5_12_3_MSVC2017_64bit-Release*
2. Acceder a la capeta llamada *release.*
3. Hacer doble click sobre el archivo *tfg.exe.* Este es el ejecutable del proyecto.

4.2 - Manuales de usuario

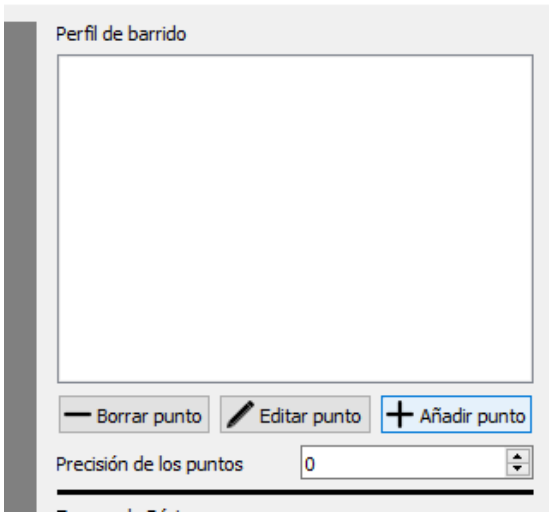
Para entender mejor el funcionamiento de la aplicación, en este documento se detalla paso a paso la funcionalidad de la misma.

Para comenzar, es necesario conocer la ventana de la aplicación.

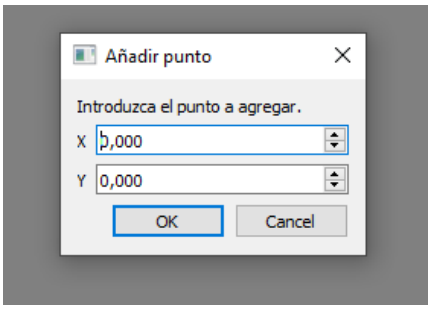


En primer lugar, es necesario que el usuario diseñe un perfil que contenga los puntos que se van a barrer. Para este paso se recomienda dibujar los puntos sobre una superficie teniendo en cuenta que el orden en el que se deben introducir más adelante es en sentido anti horario.

Una vez se tienen los puntos diseñados sobre un plano, es el momento de decidir la trayectoria por la que van a ser barridos. En este paso también se puede seguir la técnica anterior para diseñar la curva o experimentar con la herramienta que se proporciona.

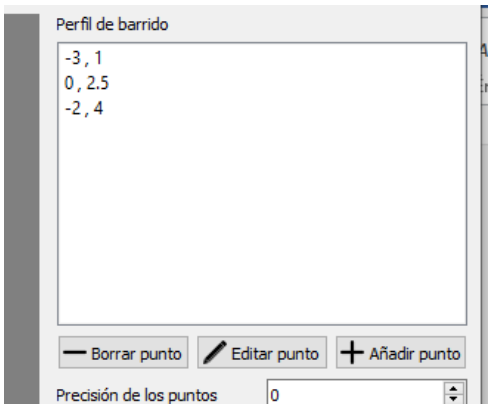


El siguiente paso es introducir los puntos que se han pensado en la aplicación. Para ello, buscamos el botón “Añadir punto” situado en la parte derecha. Una vez hacemos click sobre él, se abre una pequeña ventana.

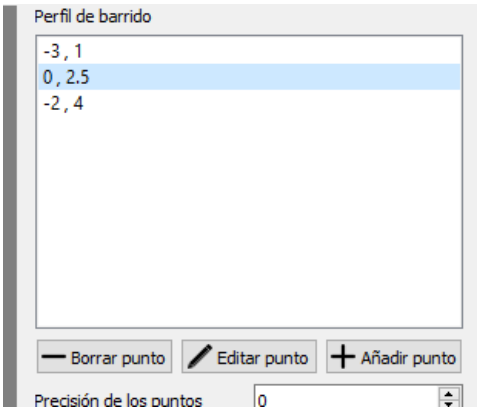


En esta ventana hay que introducir las dos coordenadas del punto que se vaya a añadir al perfil. Las coordenadas de los puntos pueden tener una precisión de hasta tres decimales y pueden tomar valores positivos y negativos. Si se quiere escribir un número decimal, se tiene que usar el carácter “,” para representar la coma del número. Una vez insertados los valores se pulsa el botón “Ok”. Para cada punto que se quiera añadir hay que realizar el mismo proceso.

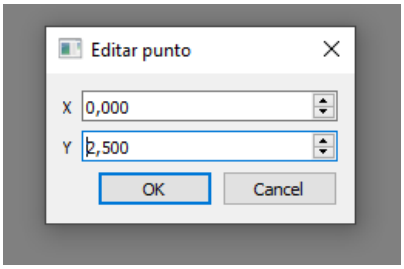
Cuando vayamos añadiendo los puntos, se irá rellenando el panel cuyo título es “Perfil de barrido”. Así se podrán visualizar los puntos que hemos insertado.



Llegados a este paso, puede ocurrir que se quiera borrar algún punto o modificar alguna de sus coordenadas. Para borrar un punto, hay que pulsar encima de él en el panel en donde aparece. Cuando esté seleccionado hay que pulsar el botón “Borrar punto”.

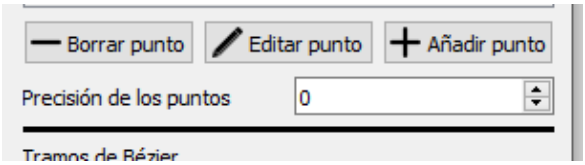


Si se quiere modificar un punto se hace lo mismo pero pulsando el botón “Editar punto”. Entonces se abrirá una ventana con las coordenadas del punto que se ha seleccionado.

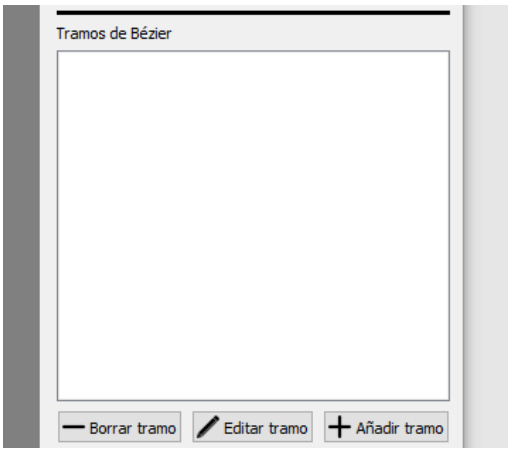


El siguiente paso es indicar si queremos que el perfil se suavice y en qué grado. Para ello hay que hacer uso del elemento llamado “Precisión de los puntos”. Por

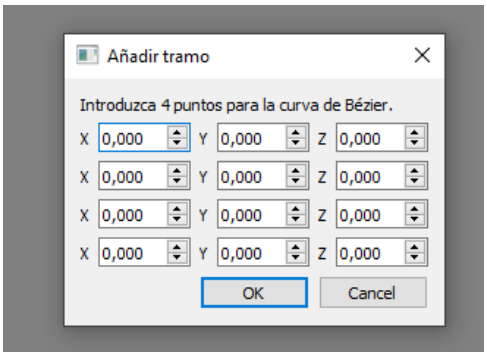
defecto está el valor 0, lo que indica que no se aplica ningún suavizado. Conforme el número sea mayor, el suavizado será mayor.



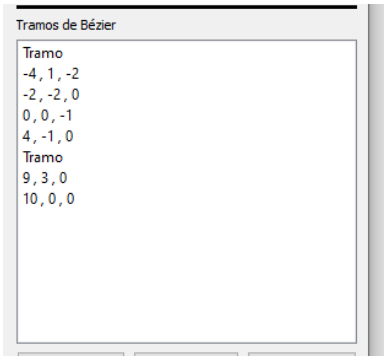
De forma análoga se introducen los datos que van a dar forma a la curva. Esta tarea se va a repartir entre el número de tramos en los que se hayan dispuesto los puntos. Se pulsa el botón “Añadir tramo” y aparece una ventana en la que se van a rellenar las coordenadas de los puntos de ese tramo.



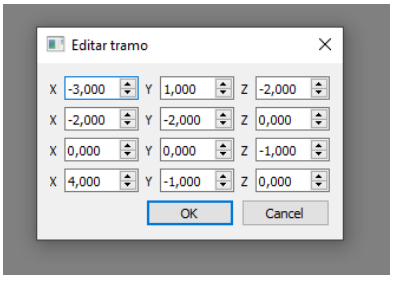
En la ventana inferior se insertan las coordenadas X, Y y Z de los puntos que van a formar el tramo. Las coordenadas de los puntos pueden tener una precisión de hasta tres decimales y pueden tomar valores positivos y negativos. Si se quiere escribir un número decimal, se tiene que usar el carácter “,” para representar la coma del número. Una vez insertados los valores se pulsa el botón “Ok”. Para cada tramo que se quiera añadir hay que realizar el mismo proceso. Cuando se inserta el primer tramo hay que añadir cuatro puntos, pero al insertar los siguientes tramos, se requieren solo dos puntos, que se corresponden con los dos últimos puntos de la curva. De esta manera los tramos quedarán unidos.



Una vez introducidos los tramos, se visualizarán en el panel titulado “Tramos de Bézier”.

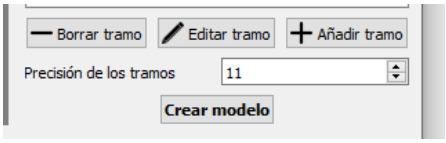


Para modificar los puntos de algún tramo o borrar algún tramo se realiza un procedimiento similar a cuando se trataba de los puntos del perfil. Primero se pincha sobre uno de los puntos del tramo para seleccionar el tramo al que pertenece y se hace click sobre “Borrar tramo” o “Editar tramo”. En el caso de modificar algún punto, se abrirá una ventana muy parecida a la que aparece cuando se quiere añadir un tramo.



A la hora de definir la curva, se puede indicar con cuanta precisión queremos que se genere, para que el modelo sea más suave y preciso. Simplemente

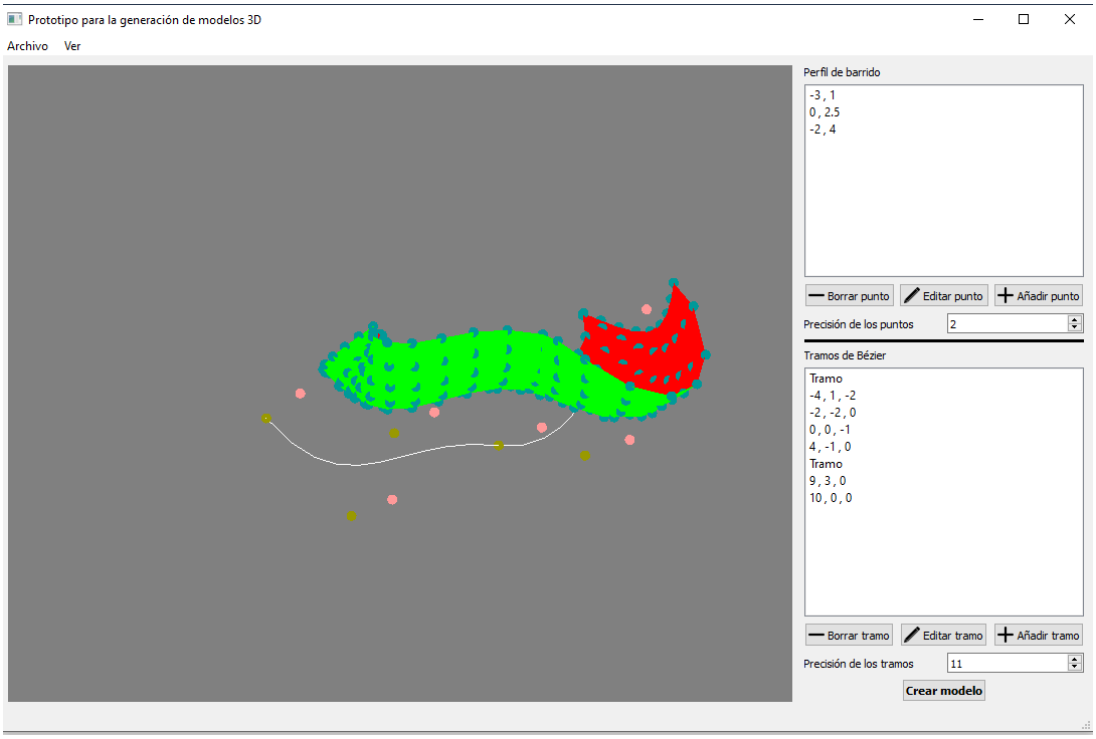
habría que modificar el campo llamado “Precisión de los tramos”. Cuanto mayor sea el valor que se introduzca, mayor será la suavidad de la curva que genera el modelo y, por tanto, el modelo será también más suave.

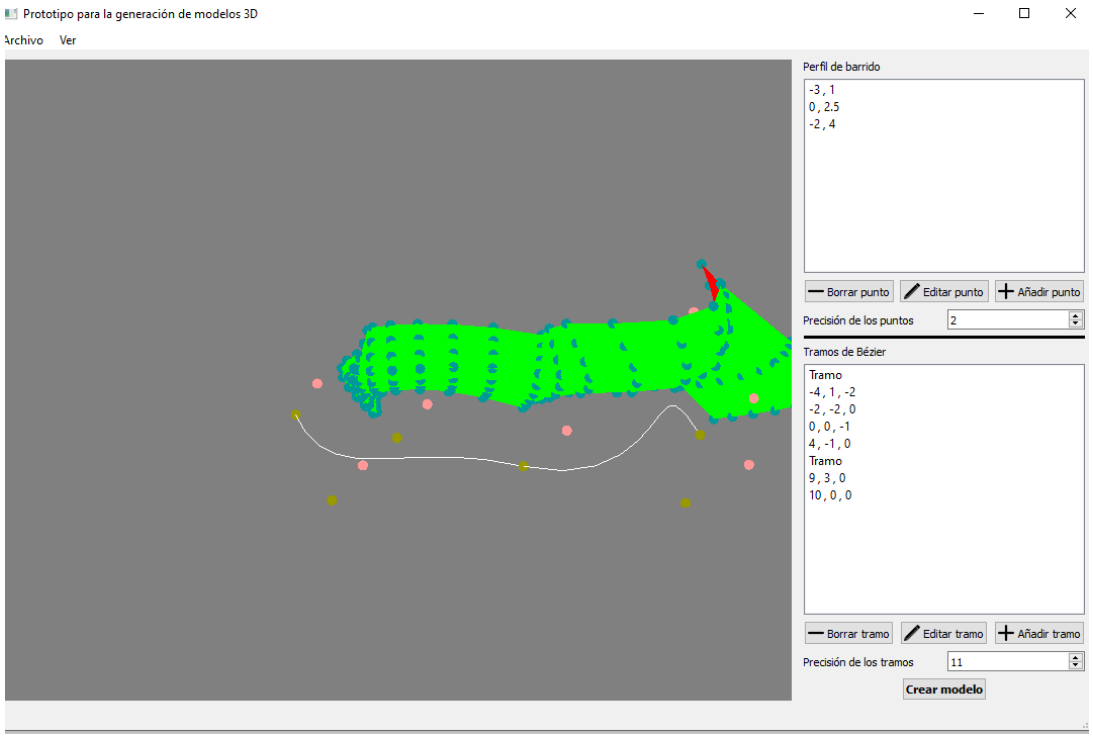


El último paso para poder visualizar los datos introducidos en forma de modelo es pulsar el botón “Crear modelo”.

En este momento se puede visualizar el modelo que hemos diseñado. La parte que se ve de color verde corresponde con la cara exterior del modelo y la parte inferior se pinta de color rojo. Los puntos azules son los que forman el modelo mediante triángulos. Los puntos ocres son los que forman la curva de cada tramo (los puntos que se han introducido para los tramos) y los rosas son puntos auxiliares que sirven para generar el modelo. También se visualiza una línea blanca que representa la unión de las curvas que definen los tramos.

Cuando se genera el modelo, la coordenada X de los puntos aparece más a la izquierda cuanto menor es y viceversa. La coordenada Y aparece más abajo cuanto menor es el valor y más arriba en caso contrario. La coordenada Z se representa con la profundidad. Estas medidas son en el momento que se genera un modelo, ya que, si nos movemos alrededor de él, cambian.



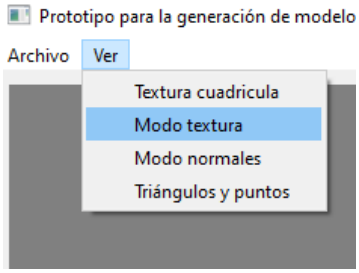


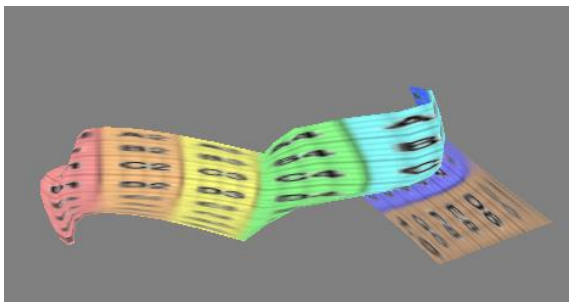
Para movernos alrededor del modelo se pincha en cualquier punto que esté en la ventana de dibujado (viene delimitada por el color gris oscuro) y se arrastra

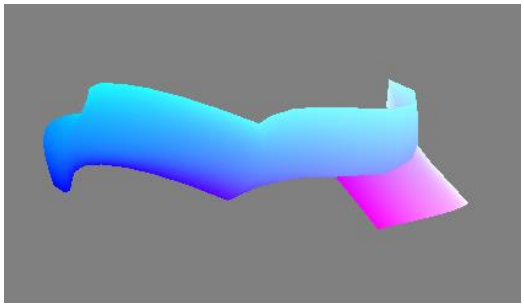
suavemente hasta conseguir la perspectiva que se desee. Además, se puede hacer zoom con el scroll.

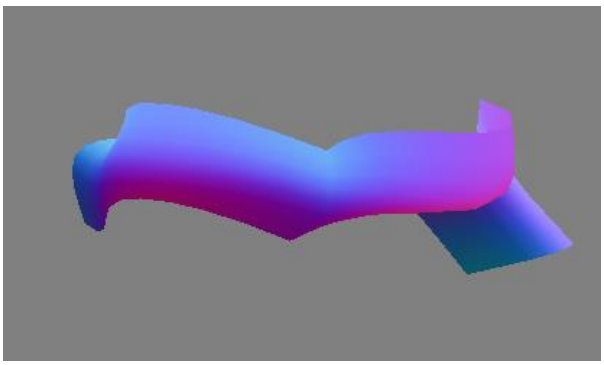
Una vez generado el modelo, podemos realizar varias acciones:

1. Visualizar otra información relevante acerca del modelo. Para ello hacemos click en la pestaña superior “Ver” y seleccionamos una de sus opciones. Así, se pueden observar las coordenadas de textura y las normales codificadas como colores y el modelo con una textura de cuadrícula.









2. Guardar la información del modelo en un archivo con formato .obj. En este se almacenarán posiciones, normales y coordenadas de textura de los vértices y la topología. Para ello se pulsa sobre la pestaña “Archivo” y “Guardar modelo”. En ese momento se abre una ventana en la que se pone nombre al archivo y se elige la ruta en la que se va a ubicar.

4.3 - Guía original del Trabajo Fin de Título

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

PROPUESTA DE TRABAJO DE FIN DE GRADO

GRADO EN:

Grado en Ingeniería Informática

ESPECIALIDAD:

General

MENCIÓN:

Sistemas Gráficos

TÍTULO DEL TRABAJO:

Prototipo para la generación de objetos 3D con topología implícita

Idioma:

Castellano

DESCRIPCION CORTA DEL TFG:

Existen diversos medios para representar objetos 3D con superficies curvadas complejas que no requieren explicitar su topología en una estructura de datos, sino que las relaciones entre los vértices que conforman los distintos triángulos de la malla se calculan matemáticamente.
Estos medios son ideales para asignaturas de programación de aplicaciones gráficas, ya que permiten a los estudiantes contar con modelos de objetos con formas complejas sobre los que probar shaders avanzados, sin necesidad de programar las estructuras de datos que los almacenan. De esta forma se pueden concentrar en los aspectos de programación de los shaders.

En la asignatura de Programación de Aplicaciones Gráficas se explica uno de esos medios, los objetos de revolución. En este TFG se propone realizar una herramienta para que los estudiantes puedan experimentar con otros medios como lofting.

El resultado de este TFG se usará en la asignatura Programación de Aplicaciones Gráficas.

Datos del Trabajo Fin de Grado:

Modalidad del TFG
Proyecto de Ingeniería

Tipo de TFG
- ○ TFG General
- ◉ TFG Específico

TFG en equipo
- ☐ TFG en Equipo

(Debe ajuntarse memoria justificativa)

Número máximo de estudiantes
1

Tutor del Trabajo Fin de Grado:

Tutor/a

Nombre:
FRANCISCO DE ASÍS CONDE RODRÍGUEZ

Departamento:
Informática

A. Conocimiento:
LENGUAJES Y SISTEMAS INFORMÁTICOS

☐ Este TFG tiene un segundo tutor

Datos del Alumno Asignado

Alumno/a asignado/a

Nombre:
María Paz Barbero Rodríguez

DNI:
77371313A

Dirección:
C/Piedras Lisas,6 Torredonjimeno (Jaén) 23650

Teléfono:
664522546

Correo-E:
mpbr0005@red.ujaen.es

Conocimientos/Requisitos Previos Recomendados

Haber cursado o estar cursando la asignatura: Programación de Aplicaciones Gráficas.

Objetivos del TFG:

- Desarrollar un prototipo de aplicación que permita:
  - Definir objetos con formas curvadas complejas mediante algoritmos que calculen implícitamente la topología.
  - Visualizar los nuevos objetos.
  - Modificar parámetros de los objetos y ver el efecto que se obtiene.
  - Salvar a disco los objetos definidos en un formato adecuado para que los estudiantes puedan incorporarlos en sus propias aplicaciones gráficas.

METODOLOGÍA A DESARROLLAR:

1. Revisión de las tecnologías disponibles, tanto en motores gráficos como en bibliotecas para la creación de interfaces gráficas.
2. Análisis del sistema, incluyendo requerimientos funcionales y no funcionales.
3. Estimación de costes y planificación temporal.
4. Diseño de la arquitectura del sistema y de las interfaces de usuario.
5. Implementación de la solución.
6. Pruebas de software.
7. Pruebas con usuarios reales (estudiantes de la asignatura PAG del prototipo).
8. Redacción de la documentación final.

DOCUMENTOS Y FORMATOS DE ENTREGA:

- Documentación generada en el proceso, de acuerdo a las buenas prácticas de Ingeniería del Software, en formato digital.
- Memoria del trabajo, incluyendo anexos y manuales de usuario e instalación, en formato digital.
- Código fuente y ejecutables.
- Documentos y archivos adicionales a los que se haga mención expresa en la memoria.
- Vídeo demostración del sistema.
- Informe favorable del tutor.
- Autorización de publicación, si procede.

DOCUMENTOS ENTREGADOS:

Documentos entregados

☑ **Solicitud de propuesta de TFG cumplimenada y firmada.**

5 - BIBLIOGRAFÍA

Pressman, R. (2010). *Ingeniería del Software.* McGraw-Hill.

Mortenson, M. (1985). *Geometric modelling.*

Conde, F. (2015). Modelado de sólidos heterogéneos mediante hiperparches

Apuntes Programación de Aplicaciones Gráficas 2018-2019

Web de Khronos

Web de OpenGL

Web de Qt

Créditos:

Iconos usados en la aplicación: web de Flaticon

Add free icon by Hanan https://www.flaticon.com/free-icon/add_109526#term=add&page=1&position=8

Remove free icon by Hanan https://www.flaticon.com/free-icon/remove_109504

Edit free icon by Hanan https://www.flaticon.com/free-icon/edit_109488