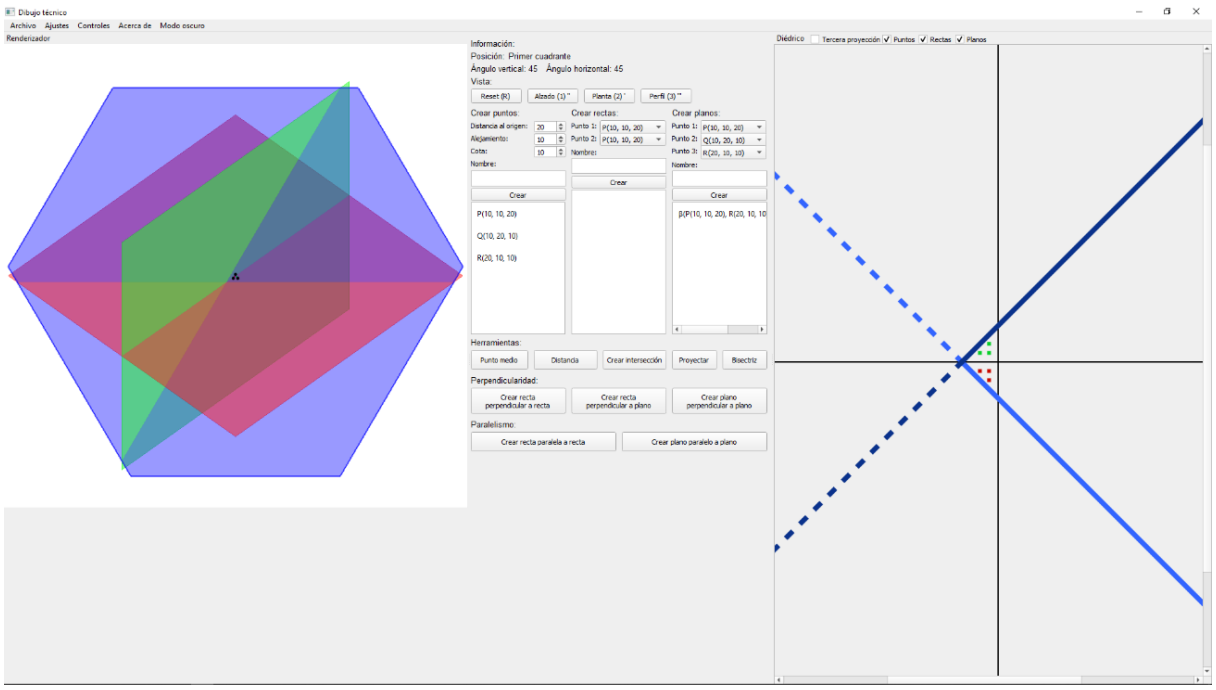
Desarrollo de un programa de dibujo técnico



Autor: Jaime Resano Aísa

Tutores: Javier Belloso Landa y José Antonio Benito Alvarado

Institución: IES Ribera del Arga

Curso 2019-2020

Agradecimientos

En primer lugar, me gustaría agradecer a los tres profesores que me han ayudado en este trabajo: Ana Isabel Esparza Ramiro, Javier Belloso Landa y José Antonio Benito Alvarado. Sin su apoyo, este trabajo no hubiera sido posible. Agradecer también la ayuda recibida por parte de programadores anónimos que han resuelto las dudas técnicas que han surgido durante el desarrollo del programa. Su labor desinteresada hace que la programación esté al alcance de cualquier persona.

RESUMEN

El dibujo técnico es un sistema de representación gráfica de diversos tipos de objetos cuyo propósito es el de proporcionar información suficiente para facilitar su análisis, modificar su diseño y posibilitar su construcción y mantenimiento (Raffino, 2019). Es frecuentemente usado en entornos industriales debido a su utilidad para poder representar precisamente objetos en un medio. Esta materia es fundamental en ciencias como la arquitectura o la ingeniería. El sistema diédrico es una rama muy importante del dibujo técnico. La escasez de herramientas informáticas enfocadas al uso educativo del sistema diédrico ha sido el principal motivo por el cual se ha desarrollado un programa sobre esta materia. En este documento se explica su diseño, funcionamiento, utilidad y limitaciones. Además, se explican las bases del sistema diédrico. Está dividido en tres partes: una parte teórica que consiste en el planteamiento de los objetivos del proyecto y una breve introducción al sistema diédrico, una parte práctica en la que se explica el funcionamiento del programa y una conclusión en la que se analiza el resultado conseguido, las limitaciones encontradas y sus posibles mejoras. También se incluyen dos videotutoriales en los que se muestra cómo instalar el programa y cómo utilizarlo.

ABSTRACT

Technical drawing is a graphical representation system of various types of objects whose purpose is to provide sufficient information to facilitate its analysis, modify its design, and enable its construction and maintenance (Raffino, 2019). It is frequently used in industrial environments due to its utility to be able to accurately represent objects in a medium. This subject is fundamental in sciences such as architecture or engineering. The multiview projection system[1] is a very important branch of technical drawing. The shortage of computer tools has been the main motivation of developing a software application about this topic. This document explains its design, operation, utility and limitations. In addition, the bases of the multiview projection system are explained. It is divided into three parts: a theoretical part consisting of the statement of the project's objectives and a brief introduction to the multiview projection system, a practical part where the behaviour of the program is explained and a conclusion in which the achieved result is analyzed and its technical limitations and possible fixes, too. This project also contains two videos which are tutorials about how to install the program and how to use it.

[1] Also known as "dihedral system"

Índice

Índice de figuras

Glosario

DIBUJO TÉCNICO	
Término	**Definición**
Arista	Recta que define un lado de un polígono.
Paralelo	Relación existente entre dos rectas o planos. Prolongando ambas entidades no se llegan a producir puntos en común.
Perpendicular	Relación que existe entre dos rectas o planos que forman cuatro ángulos de noventa grados en su intersección.
Polígono	Conjunto se segmentos unidos entre sí que delimitan un espacio cerrado.
Sistema diédrico	Método de representación de elementos geométricos en dos dimensiones.
Vértice	Punto en el que coinciden las aristas de un polígono.
PROGRAMACIÓN	
Código fuente	Conjunto de instrucciones que definen el funcionamiento de un programa.
Función	Conjunto de código que tiene un determinado propósito.
Interfaz	Conjunto de elementos gráficos de un programa o aplicación que se encuentran a la vista del usuario.
Lenguaje de programación	Conjunto de reglas que definen la forma en la que se le dan instrucciones a un sistema informático.
Python	Lenguaje de programación.
Renderizar	Proceso de dibujado de los elementos en la pantalla.
Widget	Componente de la interfaz con un determinado propósito. Por ejemplo, recibir la entrada del teclado del usuario o mostrar texto en pantalla.

Introducción

El curso educativo de bachillerato cuenta con una asignatura de dibujo técnico. Dicha asignatura se divide en tres bloques: geometría plana, sistema diédrico y representación de piezas simples. El apartado de sistema diédrico posee una alta ponderación, ya que es de vital importancia para comprender cómo funcionan los elementos geométricos en el espacio.

Existen muy pocos programas informáticos enfocados a la enseñanza. Por ejemplo, GeoGebra es una potente herramienta que se utiliza para visualizar los conceptos matemáticos que se trabajan en clase. Es altamente interactiva y fácil de usar. Sin embargo, no ocurre lo mismo para la asignatura de dibujo técnico. El material didáctico del que dispone el alumno está anticuado. Está formado por esquemas y lecciones escritas que no ofrecen ningún tipo de experiencia interactiva. Estos problemas hacen que la mayor parte del alumnado no disfrute de la asignatura. Se necesitan nuevos recursos en los que se trabaje la materia de una forma amena e interactiva.

El principal objetivo de este trabajo es el de desarrollar un programa informático que sea útil a los alumnos para comprender los ejercicios, o incluso al profesor para explicar conceptos. El programa es una herramienta, es el alumno quien debe ser capaz de utilizarlo. Está enfocado a la etapa de bachillerato, especialmente al primer año.

El programa desarrollado es interactivo. Recibe la entrada de datos del usuario y genera un resultado en forma de representación gráfica mediante un extenso conjunto de instrucciones lógicas. Cuenta con una interfaz gráfica lo más fácil de usar e intuitiva posible, ya que está principalmente enfocado al alumnado.

Cuenta con características como las de crear puntos, rectas, planos y realizar intersecciones. El objetivo final de este proyecto ha sido el de poder implementar funcionalidades como generar una recta paralela a otra, o una recta perpendicular a un plano que pase por un punto. Se han implementado el mayor número de funcionalidades posibles en la medida que el tiempo de desarrollo lo ha permitido.

Inicialmente, el propósito de este proyecto era el de explicar el funcionamiento del código del programa. Debido a la complejidad de este objetivo, se decidió no hacer esto y reducir la carga de las explicaciones. Por lo tanto, se han limitado a lo relacionado con el sistema diédrico y el uso de la aplicación.

1. SISTEMA DIÉDRICO. ASPECTOS TEÓRICOS.

El sistema diédrico[2] es un sistema de representación de elementos geométricos. Consta de dos planos perpendiculares entre sí, los cuales son superficies bidimensionales teóricamente infinitas. Sobre estos elementos se proyectan las trazas de los elementos que se dibujan, los cuales pueden ser puntos, líneas o planos.

En la intersección de ambos planos existe una recta que se conoce como línea de tierra. Esta recta se dibuja en el espacio bidimensional y sirve para posicionar los elementos existentes respecto al origen de coordenadas.

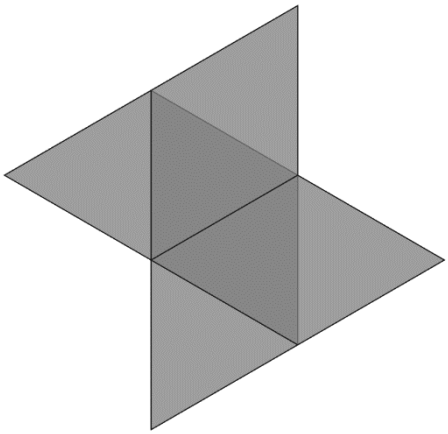


Figura 1. Planos de proyección. Elaboración propia.

Figura 2. Sistema diédrico. Elaboración propia.

Para poder establecer un sistema de coordenadas, es necesario fijar un origen. Este elemento es punto cuyo valor de sus coordenadas es nulo. Este punto se sitúa en la línea de tierra. A partir de este punto, se definen tres ejes contenidos en los planos de proyección. Cada eje muestra el valor de una coordenada, las cuales se conocen como distancia al origen, alejamiento y cota.

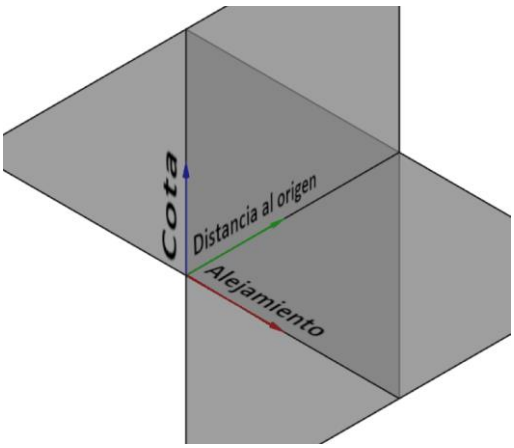


Figura 3. Ejes. Elaboración propia.

[2] El término "diédrico" tiene origen griego. Está formado por "di", dos, y "edro", cara.

1.1 Puntos

Un punto es un elemento básico del sistema. No tiene longitud ni volumen. Para localizarlo, se necesitan los valores de sus tres coordenadas respecto al origen, es decir, su distancia al origen, cota y alejamiento. El punto se representa mediante sus dos proyecciones y debe tener asignado una letra mayúscula que lo identifique.

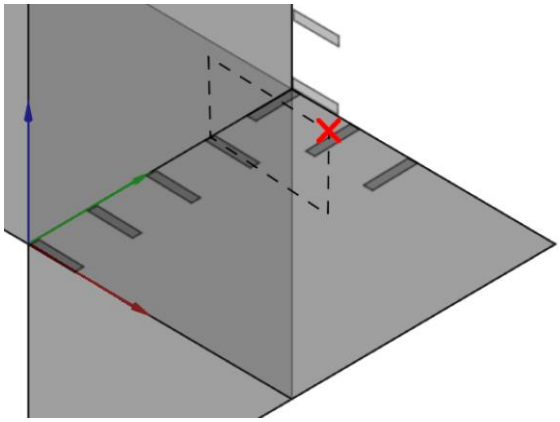


Figura 4. Punto en 3D. Elaboración propia.

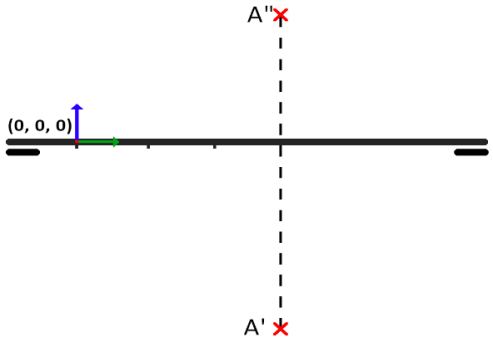


Figura 5. Punto en diédrico. Elaboración propia.

El punto representado en la imagen tiene una distancia al origen de tres unidades, un alejamiento de dos y una cota de una unidad. Su posición se puede determinar abreviadamente como "A(3, 2, 1)". Asimismo, el punto se representa en sistema diédrico mediante dos proyecciones: una asociada al plano vertical y otra asociada al horizontal[3].

Para dibujar el punto en sistema diédrico, se establece el punto de origen en la línea de tierra. A continuación, se traza una recta vertical a una distancia igual a la distancia al origen. Sobre esta recta, se dibujan dos puntos a una distancia igual a los valores de la cota y el alejamiento desde la intersección de la línea de tierra con la vertical trazada. Si la cota es positiva, se dibuja sobre la línea de tierra. Si es negativa, debajo de esta. Si el alejamiento es positivo, se dibuja debajo de la línea de tierra y viceversa.

Los planos de proyección dividen el espacio en cuatro partes llamadas cuadrantes, los cuales se numeran mediante números romanos. En la siguiente imagen están representados los dos planos proyectantes vistos de perfil.

[3] Las líneas discontinuas son auxiliares, están dibujadas para facilitar la comprensión de este ejemplo.

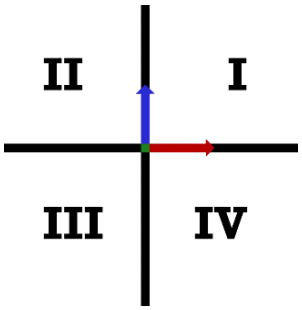
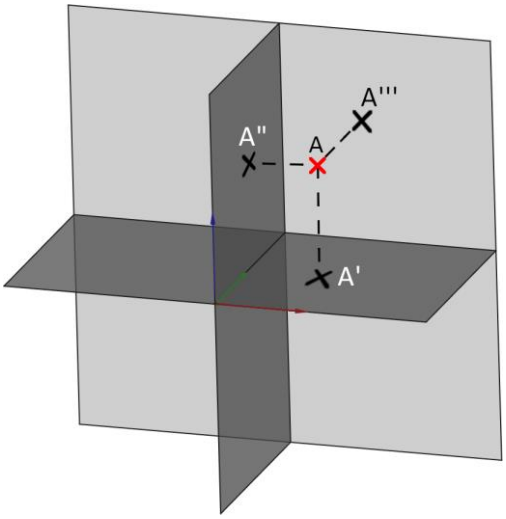


Figura 6. Cuadrantes. Elaboración propia.

Todos los puntos que se encuentren en el primer cuadrante tendrán cota y alejamiento positivo. En cambio, en el segundo cuadrante el alejamiento es negativo. En el tercer cuadrante ambos valores serán negativos y en el cuarto la cota es negativa.

Existe un tercer plano auxiliar que se conoce como plano de perfil. Este plano se puede posicionar a cualquier distancia respecto al origen y debe ser perpendicular al plano vertical y al horizontal. En sistema diédrico se representa mediante una recta perpendicular a la línea de tierra. En este plano se genera otra proyección de los elementos.



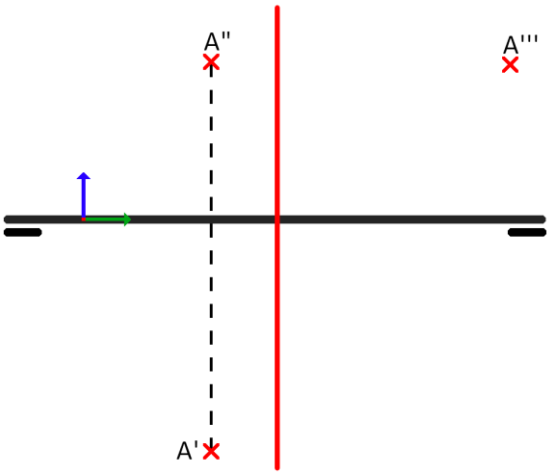


Figura 7. Tercera proyección. Elaboración propia. Figura 8. Representación diédrica. Elaboración propia.

Este plano permite obtener una tercera vista de los elementos. Es especialmente útil a la hora de representar piezas, ya que proporciona más información de los elementos dibujados.

Cabe mencionar que no existe una normalización firme sobre la nomenclatura. Para nombrar las proyecciones en el plano horizontal se añade una comilla, para las proyecciones en el plano vertical se añaden dos y a las de perfil tres.

1.2 Rectas

Una recta es una sucesión de infinitos puntos que se extienden en una dirección. Las rectas no tienen ni un punto de inicio ni uno de fin. Un segmento es una línea recta delimitada por un punto de inicio y otro de fin. Las rectas se nombran generalmente mediante letras minúsculas.

Si la recta es oblicua, corta a ambos planos de proyección. En su intersección con estos planos, aparecen dos puntos importantes. El punto contenido en el plano vertical se conoce como traza vertical (V) y el contenido en el horizontal se conoce como traza horizontal (H). Por su propia definición, el punto V siempre tiene alejamiento nulo y el punto H siempre va a tener cota nula.[4]

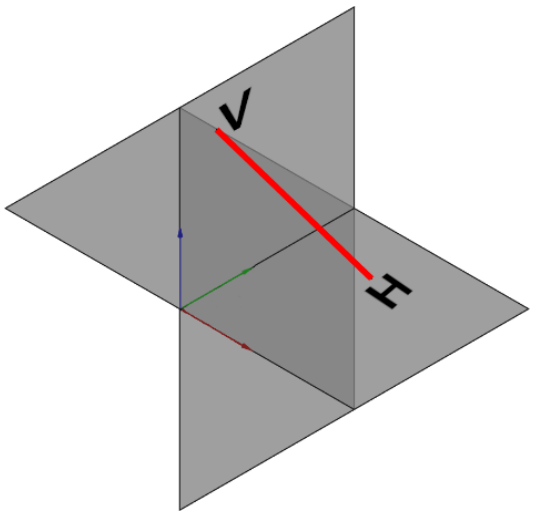


Figura 9. Recta 3D. Elaboración propia.

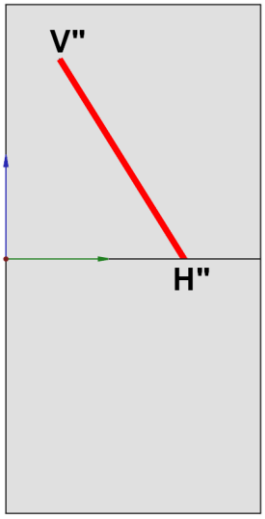


Figura 10. Vista alzado. Elaboración propia.

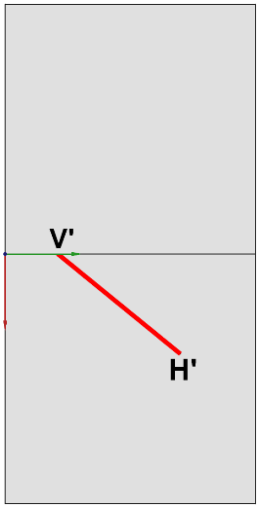


Figura 11. Vista planta. Elaboración propia.

Estas tres imágenes muestran un segmento en el espacio tridimensional, una vista del plano vertical y otra del horizontal. A continuación, se muestra la representación de una recta infinita.

[4] Los planos proyectantes son infinitos. Aparecen representados como planos finitos para lograr una mejor comprensión de la escena. Las figuras Figura 10 y Figura 11 son imágenes tomadas desde distintas posiciones del observador. Para orientarse es recomendable tener en cuenta la posición de los ejes.

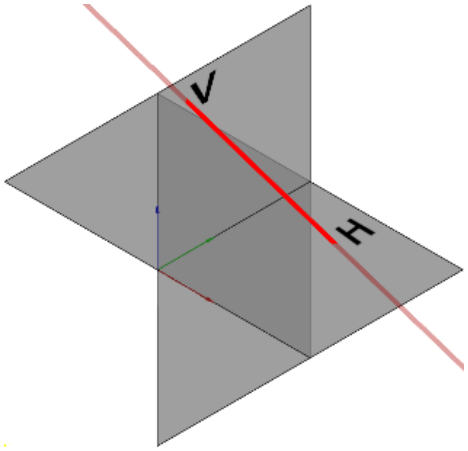


Figura 12. Recta infinita. Elaboración propia.

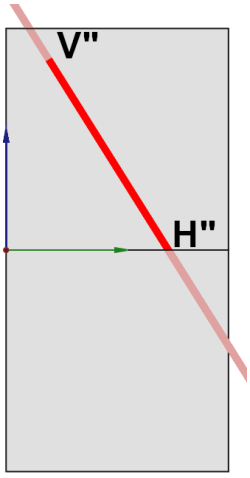


Figura 13. Alzado. Elaboración propia.

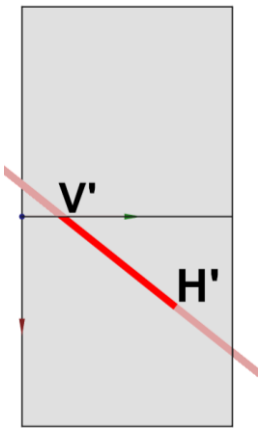


Figura 14. Planta. Elaboración propia.

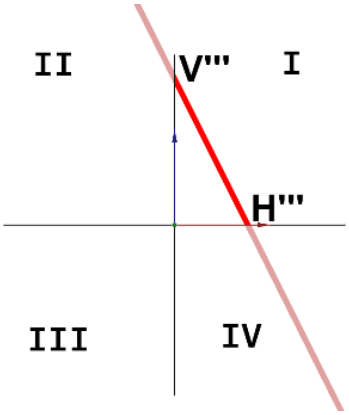


Figura 15. Perfil. Elaboración propia.

Dicha recta tiene su representación en el sistema diédrico, la cual se muestra en la siguiente figura.

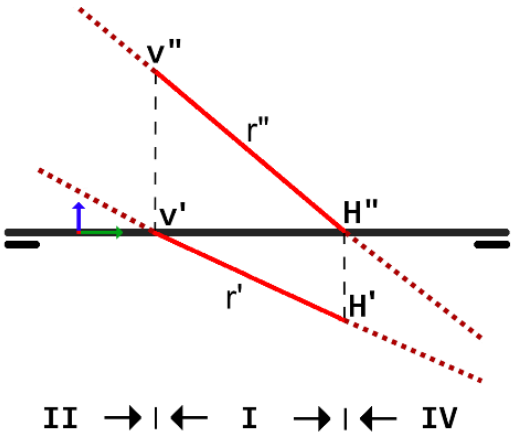


Figura 16. Recta infinita. Elaboración propia.

La recta r, al igual que el punto, posee dos proyecciones asociadas. Cada punto de la recta r" tiene un punto asociado en la recta r' y viceversa. Esta recta atraviesa tres cuadrantes. Los elementos geométricos sólo se consideran visible cuando se encuentra en el primer cuadrante. Cuando la recta no es visible se dibuja mediante líneas discontinuas.

1.3 Planos

Un plano es una superficie infinitamente fina que contiene infinitos puntos y rectas. En sistema diédrico se representa mediante las rectas que forma su intersección con los planos proyectantes. Los planos generalmente se denominan mediante letras griegas[5].

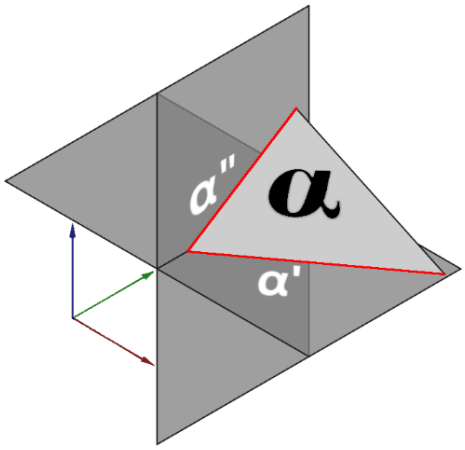


Figura 17. Plano. Elaboración propia.

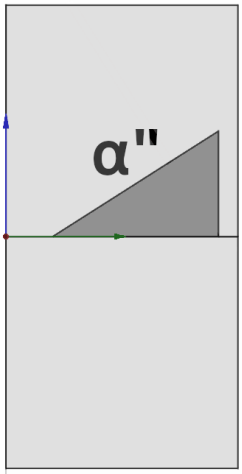


Figura 18. Alzado. Elaboración propia.

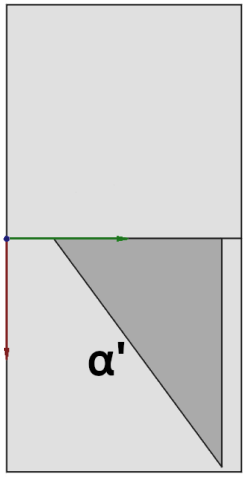


Figura 19. Planta. Elaboración propia.

A las rectas que se encuentran en la intersección del plano con los planos proyectantes se les asigna el nombre del plano. A la traza vertical se le asignan dos comillas y a la horizontal una. En la siguiente figura se muestra el plano en sistema diédrico:

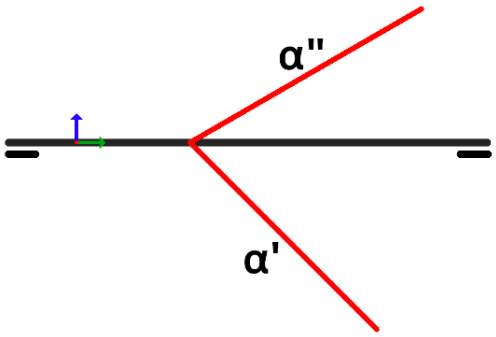


Figura 20. Representación del plano. Elaboración propia.

[5] En las figuras 18 y 19 se representan las partes del plano de ejemplo pertenecientes al primer cuadrante. El plano continúa infinitamente atravesando todos los cuadrantes.

2. DESARROLLO. PARTE PRÁCTICA.

Antes de comenzar a programar, es necesario analizar el problema y fijar unos objetivos claros sobre lo que se pretende lograr. La idea inicial es crear un programa dividido en tres partes: un visor de las tres dimensiones del espacio con los planos proyectantes, un visor de los elementos geométricos en sistema diédrico y un panel con diferentes opciones que permita crear puntos, rectas y planos.

El lenguaje de programación elegido para el programa ha sido Python. Este lenguaje es fácil de usar y tiene una curva de aprendizaje muy apta para programadores novatos. Su gran popularidad ha hecho que se hayan desarrollado una gran cantidad de librerías que proveen un gran número de funcionalidades. Se ha decidido utilizar el estándar OpenGL para el dibujado tridimensional de los elementos.

El código fuente del programa se guarda en un solo archivo. Se ha utilizado un sistema de control de versiones llamado Git para guardar registro de la evolución del código. Es muy popular entre los desarrolladores debido a su alta utilidad y a la existencia de una versión de alojamiento web del código llamada GitHub. Cualquier persona que cuente con los conocimientos necesarios puede contribuir al desarrollo del programa, añadiendo nuevas funciones o corrigiendo los posibles errores que hayan pasado desapercibidos. El repositorio en el que se guarda el código está disponible en el siguiente enlace[6]:

Figura 21. Código QR que enlaza al repositorio. Elaboración propia.

[6] https://github.com/Jaime02/Proyecto-de-investigacion-2019-Dibujo-tecnico

2.1 Interfaz

La interfaz que el usuario ve y utiliza del programa. Sirve para recibir la entrada de datos y mostrar los resultados en pantalla.

Figura 22. Boceto inicial de la interfaz. Elaboración propia.

Durante el proceso de desarrollo, la interfaz se convirtió en un factor limitante para la implementación de nuevas funciones. Esto supuso que se diseñara una nueva desde cero teniendo en cuenta los problemas de la anterior. Cabe destacar que algunas de las imágenes de este documento pertenecen a etapas en las que la interfaz no había sido totalmente finalizada, pero esto no afecta en absoluto a la comprensión de lo explicado.

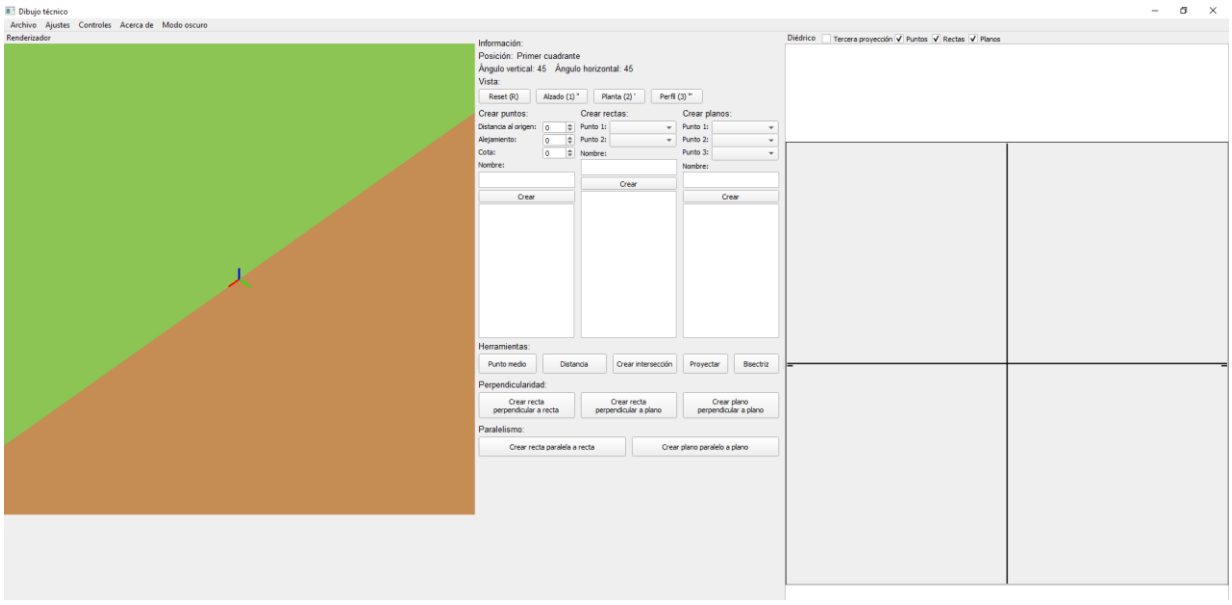
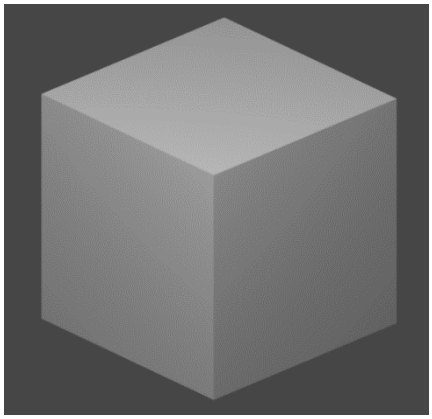


Figura 23. Interfaz final. Elaboración propia.

2.2 Cámara

La perspectiva define la forma en la que se ven los objetos. Existen dos tipos de perspectivas: la perspectiva cónica y la ortogonal. La perspectiva cónica es la que posee el ser humano. Existen puntos de fuga en el infinito en los que las líneas paralelas se juntan. En cambio, en perspectiva ortogonal las rectas paralelas siempre se ven paralelas.



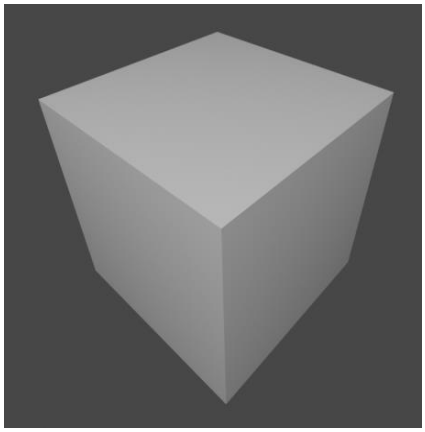


Figura 24. Perspectiva ortogonal. Elaboración propia. Figura 25. Perspectiva cónica. Elaboración propia.

La perspectiva cónica es más realista. Sin embargo, la información que percibe el usuario debe ser lo más precisa posible, por lo que se ha escogido la ortogonal.

Posteriormente, se diseñó un sistema para mover la cámara y permitir que el usuario se mueva alrededor del espacio. La investigación sobre cómo implementar una cámara que el usuario pueda mover y rotar ha sido extensa. Tras realizar una búsqueda exhaustiva en internet y preguntar en foros dedicados a la programación como Stack Overflow, se decidió implementar una cámara esférica que rote sobre un punto central.

$$\begin{cases} x &= r\sin\theta\ \cos\varphi \\ y &= r\sin\theta\ \sin\varphi \\ z &= r\cos\theta \end{cases}$$

Figura 26. Coordenadas. Fuente: Wikipedia.com.

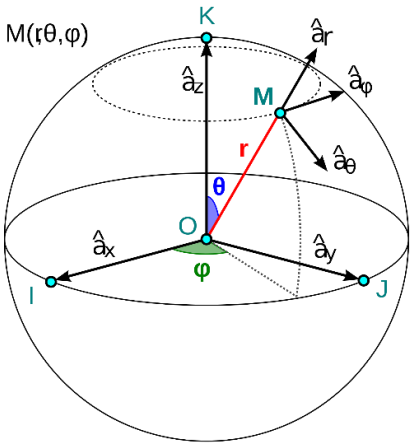


Figura 27. Esfera. Fuente: Wikipedia.com

2.3 Puntos

Tras conseguir una interfaz robusta, se implementaron los puntos. El usuario debe introducir los valores de la distancia al origen, alejamiento y cota del punto que va a ser creado. También le puede asignar un nombre. En caso de que no lo haga, se le asignará una letra mayúscula.

El color elegido para representar el alejamiento del punto es el rojo, ya que al observarlo desde arriba se visualiza el plano horizontal, el cual es rojo. En el caso de la cota ocurre lo mismo, pero con el color verde. La tercera proyección de los elementos se dibuja en color negro.

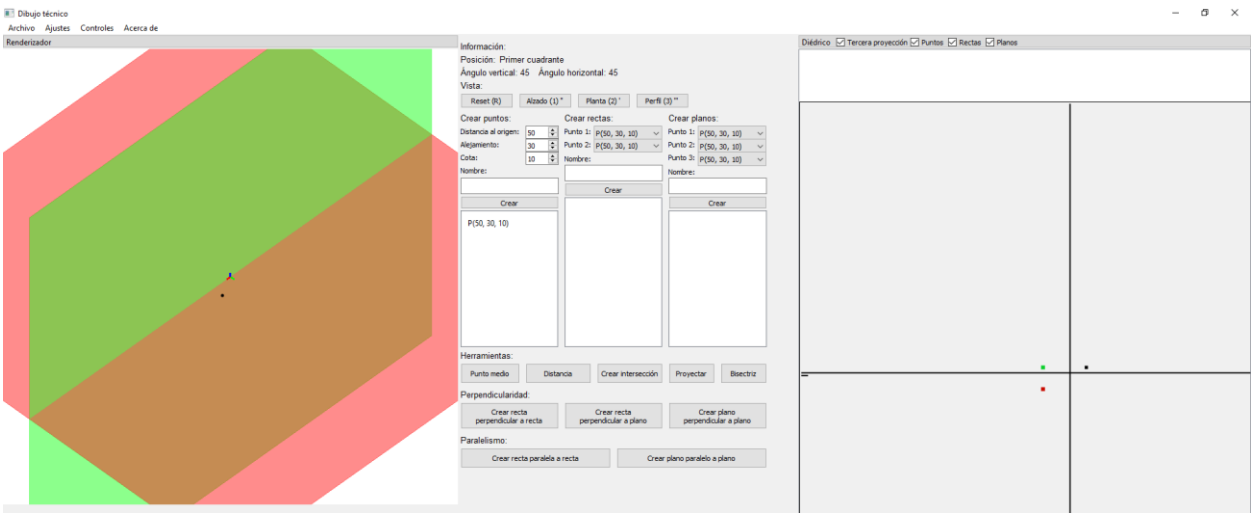


Figura 28. Punto. Elaboración propia.

2.4 Rectas

El proceso de crear puntos no ha presentado grandes dificultades. En cambio, la programación de la recta ha sido bastante más compleja debido a su propia naturaleza. Una recta es teóricamente infinita. Sin embargo, OpenGL solo permite dibujar segmentos, es decir, líneas finitas con dos extremos.

Una recta no puede ser creada a partir de dos puntos coincidentes, este es el único caso en el que se produce un error. Para evitar que esto ocurra, se comprueba que los valores de las coordenadas de los puntos no sean iguales. Cuando se da este caso, se muestra un mensaje de error y se cancela la creación de la recta.

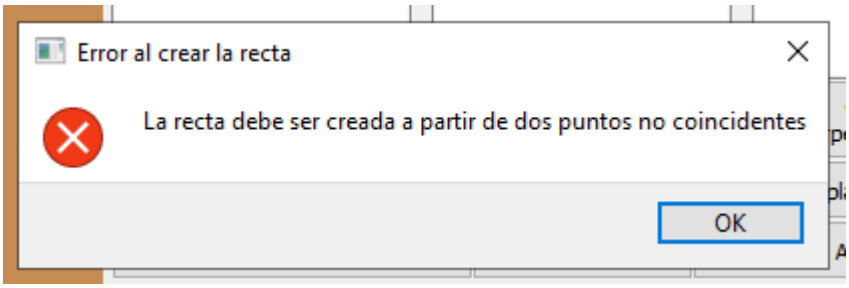


Figura 29. Error al crear la recta. Elaboración propia.

Se utiliza el módulo Sympy Para calcular los extremos de la recta. Sympy es una librería que permite utilizar matemática simbólica. Se pueden realizar cálculos algebraicos avanzados mediante las herramientas que provee. También contiene funciones geométricas que permiten crear puntos, rectas y planos y realizar operaciones entre estos elementos.

El algoritmo define las seis caras del cubo ficticio de 1000 unidades de lado en el que se encuentran contenidos todos los elementos dibujados. No se pueden crear puntos, rectas o planos fuera de este espacio. Cada cara del cubo es un plano. Posteriormente, se realizan intersecciones de la recta con todas las caras. Si la coordenada del punto de intersección está en el rango entre quinientas y menos quinientas unidades, la recta atraviesa a esa cara en ese punto. En caso de que los valores no estén en ese rango, la recta ha atravesado otra cara antes de llegar a la probada[7].

[7] Nótese que para simplificar la explicación del algoritmo las imágenes muestran un cuadrado en lugar de un cubo.

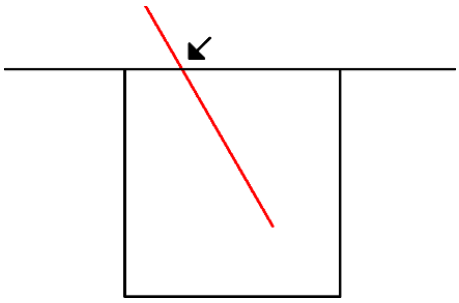


Figura 30. Intersección bien. Elaboración propia.

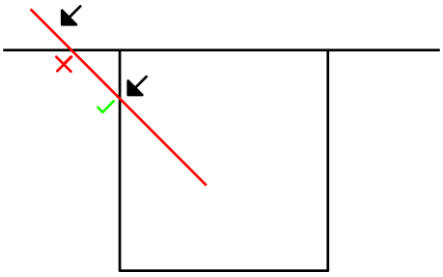


Figura 31. Intersección mal. Elaboración propia.

En la primera imagen, la cara seleccionada es la que primero es atravesada por la recta, a diferencia de en la segunda figura. Las coordenadas del punto generado en el primer caso estarán en el rango deseado, sin embargo, en el segundo no lo están. Por lo tanto, el punto que se utiliza es el de la cara del primer caso. Este proceso se realiza con todas las caras del cubo. Así se obtienen los dos extremos del segmento deseado. Debido al uso de planos, el algoritmo no es eficiente y, dependiendo de las características del equipo en el que se utiliza el programa, el tiempo de ejecución de este puede llegar a ser percibido por parte del usuario.

Antes de comenzar a desarrollar el algoritmo, es recomendable crear un diagrama de flujo que muestre el proceso de decisión necesario para dibujar una recta.

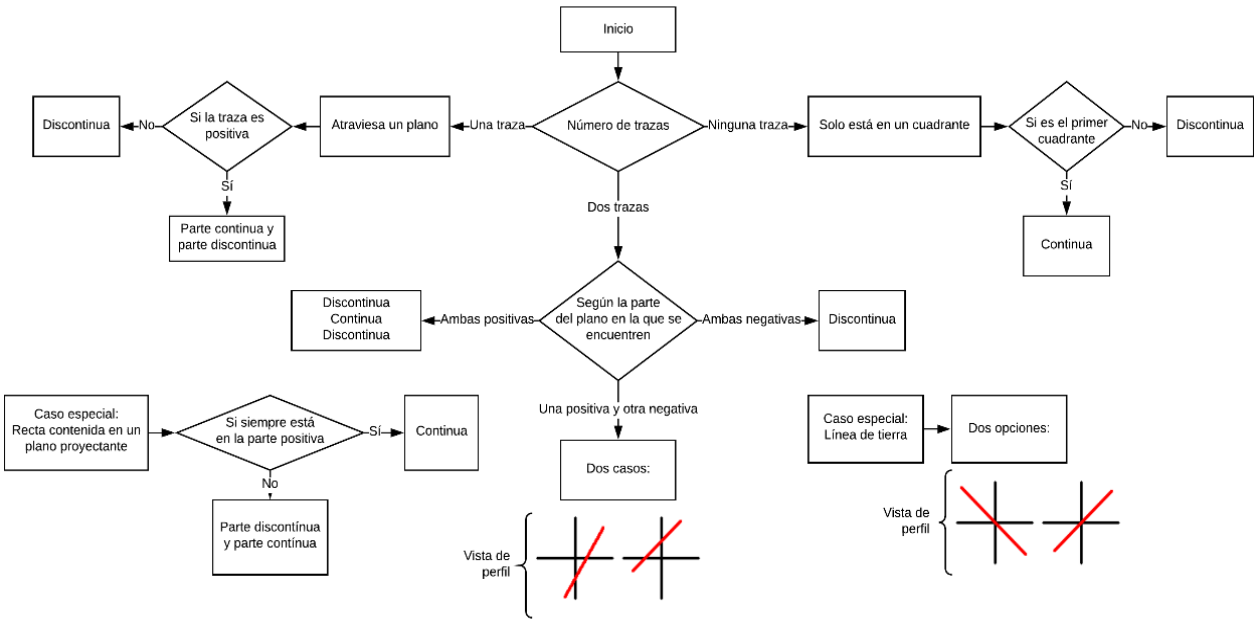


Diagrama 1. Diagrama de las rectas. Elaboración propia.

El diagrama está compuesto por una rama principal, la cual se divide en tres opciones dependiendo del número de trazas. Existen dos casos especiales, representados fuera de la rama principal. El diseño del algoritmo ha sido costoso en cuanto a tiempo debido a la cantidad de posiciones de rectas que existen.

Cuando dos elementos se superponen, el elemento dibujado más recientemente eclipsa al primero. Se intentó hacer que los colores se mezclaran al superponerse para mejorar la visualización en estos casos, pero los resultados no fueron lo suficientemente buenos como para que esta característica fuera implementada.

Llegados a este punto, las principales funciones respecto a puntos y rectas han sido creadas, pero el tiempo de desarrollo ha sido excesivo. El código de todo el programa ha sufrido un gran número de cambios debido a que inicialmente se desconocían los problemas a los que se iba a hacer frente.

2.5 Planos

El último elemento geométrico fundamental que ha sido implementado es el plano. Se puede crear un plano mediante tres puntos no alineados o coincidentes.

El algoritmo de intersección del plano con el cubo ha sido significativamente más complejo que el de las rectas debido a la cantidad de diferentes casos que pueden surgir. El principal problema es la cantidad de vértices que se generan en la intersección del plano con el cubo ficticio que se utiliza para el renderizado. El número de vértices generados puede ser tres, cuatro, cinco o seis.

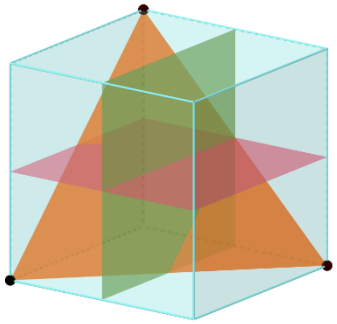


Figura 32. Plano con tres vértices. Elaboración propia.

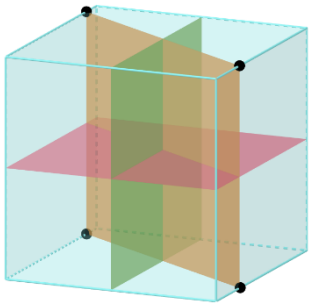


Figura 33. Plano con cuatro vértices. Elaboración propia.

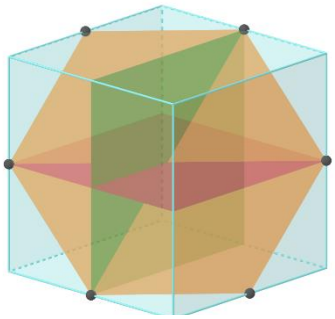
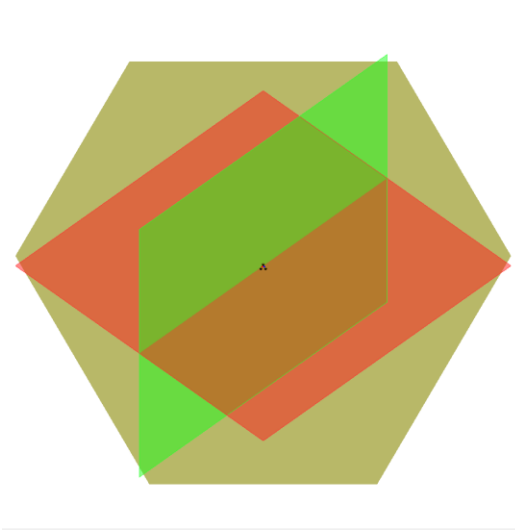


Figura 34. Plano con seis vértices. Elaboración propia.

Se necesitan dos puntos para definir y posteriormente renderizar una recta. En cambio, para dibujar un plano se utilizan los puntos de las aristas del cubo generados a partir de la intersección del plano con estas. Se encuentran representados en negro en las tres figuras superiores.

OpenGL permite dibujar triángulos y cuadriláteros. En caso de que el plano se represente como un triángulo no existe ningún inconveniente, ya que se dibujan los tres vértices sin tener en cuenta su orden y OpenGL dibuja el triángulo correctamente. Sin embargo, cuando hay más de tres vértices, el orden en el que se dibujan afecta al resultado. Esto puede producir resultados incorrectos, como que se unan dos vértices no consecutivos y no se obtenga el resultado deseado. Las siguientes imágenes son ejemplos de este problema. En las figuras 36 y 38 los planos han sido dibujados con sus vértices desordenados. En el momento en el que se tomaron las capturas del programa, las transparencias no funcionaban correctamente, por lo que no se aprecian las trazas del plano.



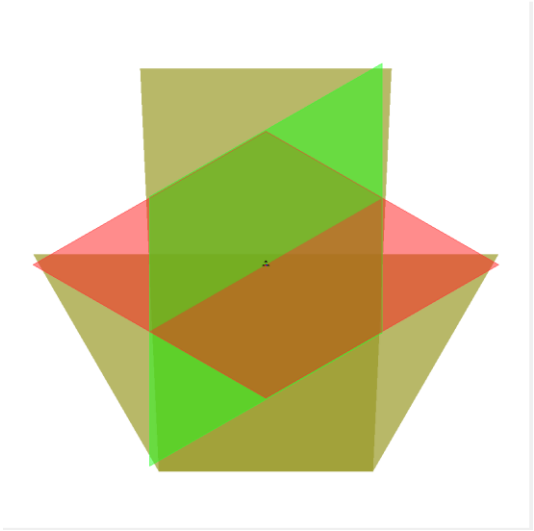
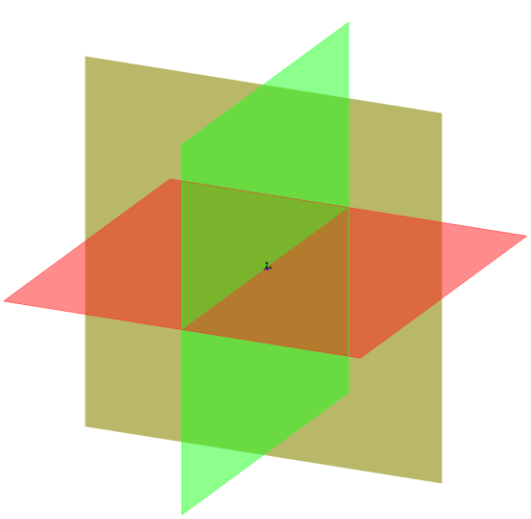


Figura 35. Plano hexágono bien. Elaboración propia. Figura 36. Plano hexágono mal. Elaboración propia.



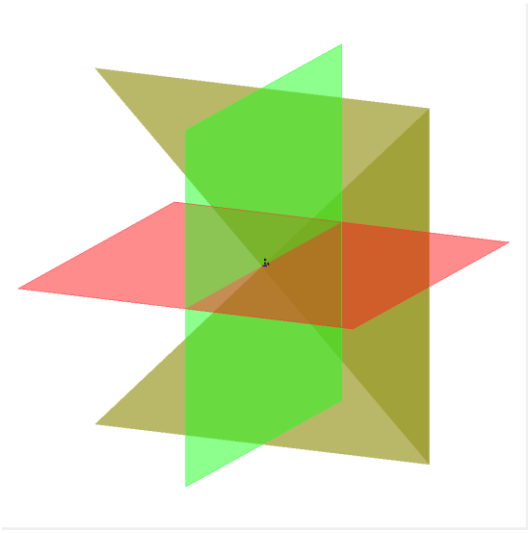


Figura 37. Plano de perfil bien. Elaboración propia. Figura 38. Plano de perfil bien. Elaboración propia.

Para solventar este grave problema, se investigó sobre algoritmos que permitieran ordenar correctamente los vértices. Tras una serie de fallidos intentos, se encontró una solución. Consiste en proyectar los vértices del plano sobre otro plano para poder ordenarlos y calcular el ángulo que forman respecto al origen mediante la función matemática arcotangente. Esta función devuelve el ángulo en radianes que forma un punto respecto al origen. Finalmente se ordenan de mayor ángulo a menor, aunque si fuera de menor a mayor el polígono se dibujaría igualmente.

El plano sobre el que se proyectan los puntos no siempre es el mismo. Esto se debe a que, por ejemplo, si se proyectan sobre un plano paralelo al plano horizontal y el plano que se pretende crear es perpendicular a este, los vértices proyectados sobre el plano horizontal coincidirán y no podrán ser ordenados correctamente.

El siguiente paso ha sido mejorar la translucidez de los elementos, ya que en las figuras 35 y 37 no se aprecia qué elemento está detrás de otro. Tras investigar sobre si OpenGL permite realizar esto automáticamente, se llegó a la conclusión de que la única solución es dividir todos los elementos en cuadrantes para poder dibujarlos en orden, siendo los primeros los que se encuentren más alejados de la cámara y los últimos los más cercanos. Por lo tanto, el orden en el que se dibujan los elementos varía dependiendo de la posición del observador. Por ejemplo, si el observador se encuentra en el primer cuadrante, primero se dibujan los elementos que estén en el cuarto cuadrante, posteriormente se dibuja la zona negativa del plano horizontal y la del plano vertical. A continuación, se dibujan los elementos del segundo y cuarto cuadrante, la zona positiva del plano horizontal, la del plano vertical y los elementos del primer cuadrante. Esto evita solapamientos que producen que la translucidez no funcione correctamente, como ocurre en las figuras 35 y 37.

Para conseguir este objetivo, se tuvieron que rediseñar completamente los algoritmos de la recta y el plano. La recta se dividió en partes, según los cuadrantes que atravesaba. Además, se añadió la opción de crear segmentos en lugar de rectas. Esta característica requirió el desarrollo de otra versión del algoritmo de segmentación de las rectas aún más compleja, ya que el anteriormente diseñado no era apto para dibujar segmentos.

Cabe mencionar que los algoritmos para dibujar los elementos geométricos en tres dimensiones y en sistema diédrico son totalmente diferentes. Las siguientes figuras muestran un ejemplo de una recta y su correspondiente segmento correctamente dibujados.

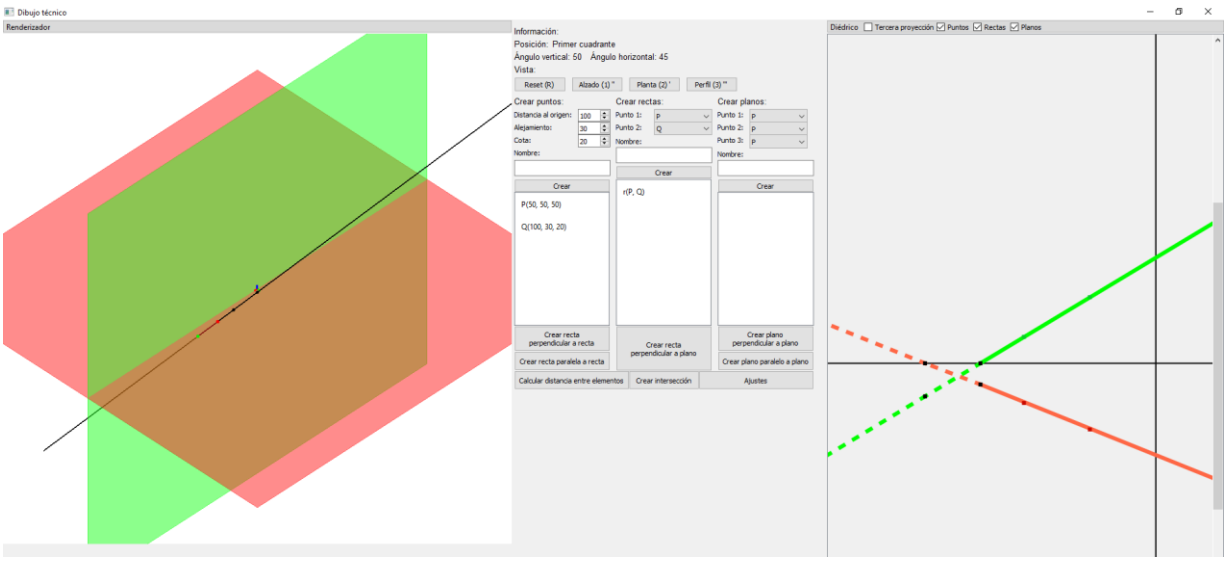


Figura 39. Recta. Elaboración propia.

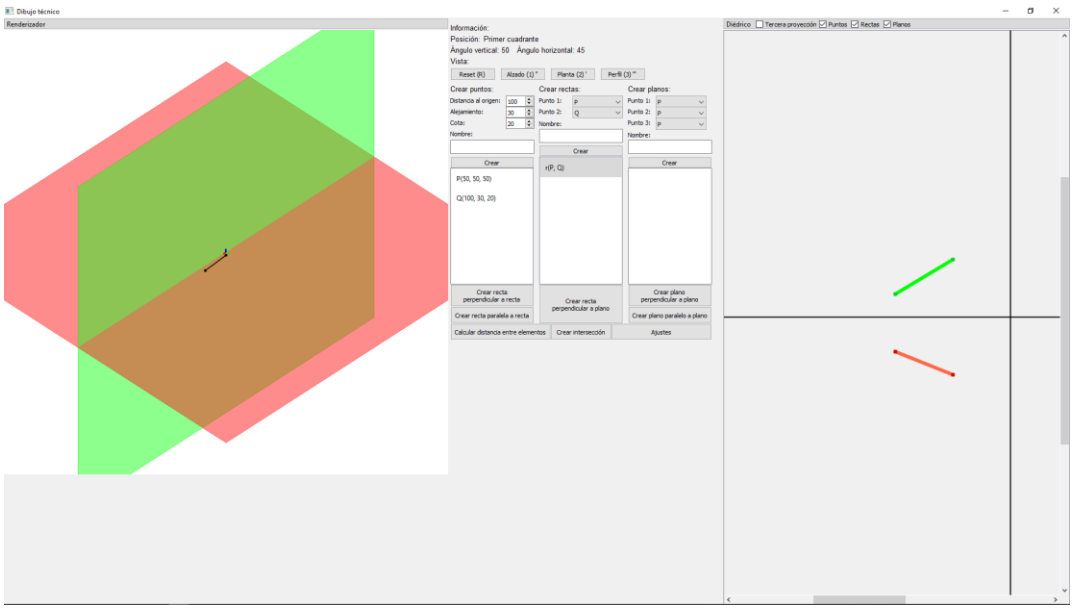


Figura 40. Segmento. Elaboración propia.

Para los planos se realizó el mismo proceso. Se diseñó un algoritmo que divida el plano entre los cuadrantes por los que pasa y otro que dibujara sus partes en orden. Se intentaron crear opciones para los planos finitos, pero fueron parcialmente desarrolladas debido a la alta dificultad de alcanzar una solución óptima. El principal problema reside en que el plano finito definido por tres puntos es un triángulo. Este triángulo debe ser correctamente dividido entre los cuadrantes que atraviesa para que pueda ser dibujado en orden. Existe una altísima cantidad de posibles casos que dependen del cuadrante en el que residan los puntos, por lo que no se ha logrado implementar un algoritmo que sea efectivo. La siguiente imagen muestra un plano oblicuo correctamente dibujado.

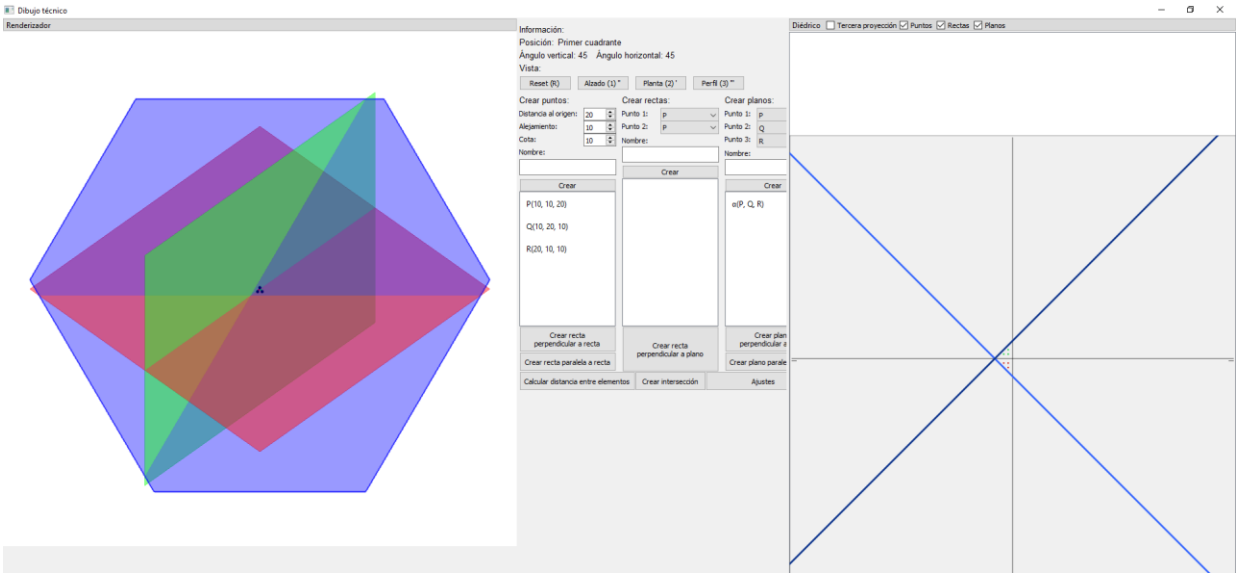


Figura 41. Plano. Elaboración propia.

El proceso de dibujado de las trazas de los planos en sistema diédrico fue bastante trivial, ya que son dos rectas fáciles de dibujar. Los colores que se utilizan en el *widget* del sistema diédrico no son modificables, a diferencia de los del *widget* del espacio tridimensional.

En la etapa final del desarrollo se añadieron diferentes características que permiten el paralelismo y la perpendicularidad entre elementos, así como la posibilidad de realizar intersecciones y calcular distancias. La librería Sympy ha sido de vital importancia para conseguir estas funcionalidades.

Además, se creó un pequeño apartado de ajustes, el cual permite activar y desactivar ciertas opciones que pueden ser de utilidad a usuarios avanzados.

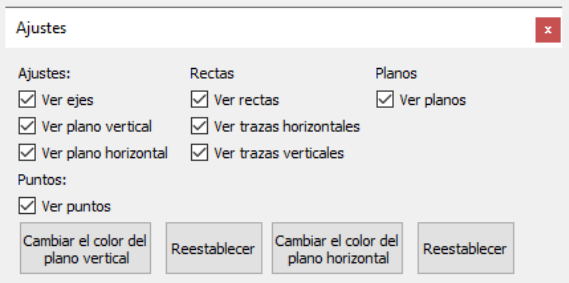


Figura 42. Ajustes. Elaboración propia.

Finalmente, se creó un sistema de guardado de archivos que permite tanto guardar como cargar los elementos creados. Este es el aspecto final de la interfaz del programa:

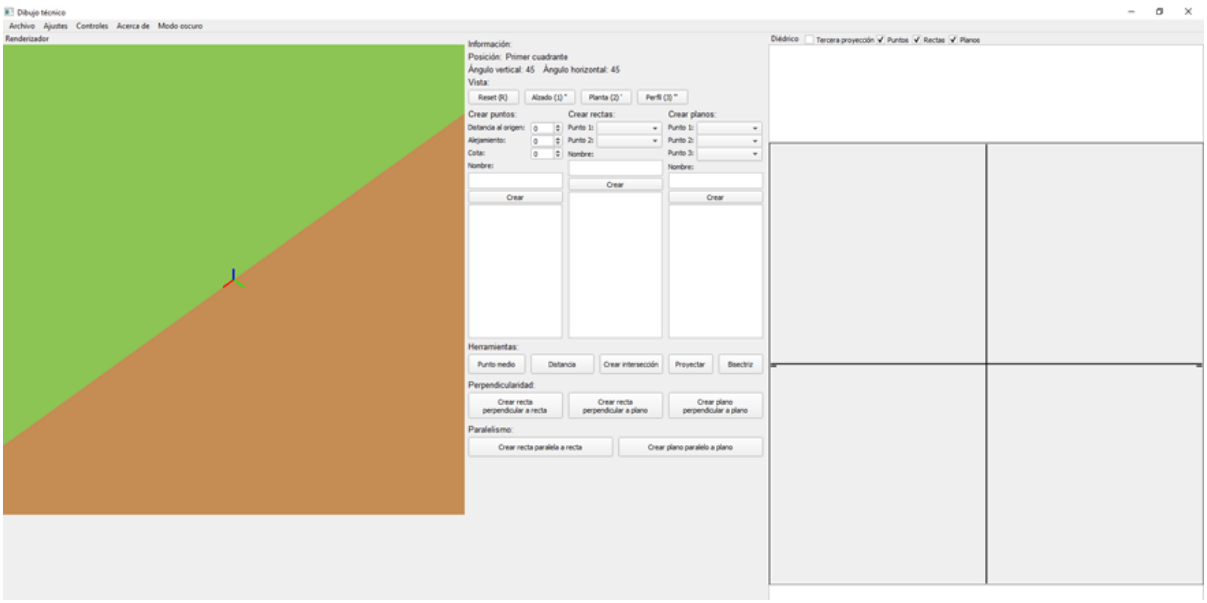


Figura 43. Interfaz final. Elaboración propia.

Aquí concluye el desarrollo del programa, el cual comenzó en agosto de 2019 y finalizó en marzo de 2020.

3. CONCLUSIÓN

El desarrollo de la aplicación ha concluido satisfactoriamente. El programa cuenta con una gran variedad de características que son de gran utilidad a la hora de realizar ejercicios de dibujo técnico.

El código desarrollado cuenta con unas 2800 líneas. Es extenso y complejo, pero está escrito conforme a las guías y convenciones propias del lenguaje de programación. Su legibilidad podría ser mejorada para que aquellos programadores que lo lean lo comprendan con mayor facilidad.

El proceso de desarrollo ha durado aproximadamente siete meses. El tiempo dedicado a la programación no ha sido medido, pero se estima que ha sido de unas quinientas horas. Esta cifra tan elevada se debe a que la mayor parte del tiempo se ha estado investigando cómo se pueden conseguir los objetivos en lugar de escribiendo el código. Durante el proceso de la programación, la mayor parte del tiempo se invierte en la lectura de código y la búsqueda de información. Esta etapa ha sido caótica debido a la falta inicial de conocimientos. Esto ha provocado que gran parte del código inicial fuera reescrito. Por ejemplo, la forma en la que se dibujan las rectas ha sido reescrita tres veces, al igual que la de los planos.

Los objetivos eran ambiciosos y se desconocía si podían ser cumplidos. La mayor parte de ellos han sido satisfactoriamente alcanzados. Sin embargo, existen ciertos apartados que podrían ser mejorados.

3.1 Posibles mejoras

Una de las principales limitaciones del programa es que el espacio en el que se pueden crear los elementos geométricos está limitado a un cubo de mil píxeles de largo, alto y ancho. Esta limitación se debe al propio funcionamiento interno del programa. Se podría aumentar este límite considerablemente, pero el resultado obtenido no sería óptimo debido a las limitaciones de la interfaz.

La transparencia de los elementos dibujados ha sido, sin lugar a dudas, uno de los retos más complejos que han aparecido. Finalmente, ha podido ser correctamente desarrollada, pero el código que permite esta funcionalidad es bastante complejo en cuanto a extensión y diseño. Esto puede suponer que sea difícil su comprensión.

El plano de perfil ha sido situado en el origen de coordenadas. No puede ser movido por el usuario. Existen varias alternativas a este problema, pero no ha sido una cuestión prioritaria, por lo que no se ha añadido la posibilidad de moverlo.

El rendimiento del programa en cuanto a renderizado puede llegar a ser sorprendentemente malo si se dibujan muchos elementos. Las causas de este problema probablemente sean que se han utilizado funciones obsoletas de OpenGL. Los elementos dibujados son muy simples, no deberían generar ningún tipo de problema, pero pueden llegar a notarse bajadas de rendimiento, especialmente en ordenadores poco potentes.

Todos los elementos dibujados son finitos. Este problema parece imposible de resolver. La única solución posible consiste en dibujarlos tan lejos del observador que parezcan infinitos.

Cuando hay varios elementos creados, puede llegar a ser confuso identificar a qué elemento corresponde cada dibujo. Esto podría solucionarse detectando sobre qué elemento está el ratón y mostrando su nombre en pantalla. No se ha investigado cómo se podría implementar esta funcionalidad debido a la falta de tiempo.

Se han desarrollado satisfactoriamente la gran mayoría de funciones necesarias para la etapa de primer de bachillerato. Sin embargo, en segundo de bachillerato se introducen elementos geométricos complejos como tetraedros, octaedros, conos, cubos o pirámides. Los retos matemáticos que surgen para lograr la adición de estas entidades son muy interesantes. Debido a la falta de tiempo y a las propias limitaciones de la interfaz no se han logrado implementar ninguna de estas entidades geométricas.

Otro problema existente es el lenguaje de la interfaz del programa. Todos los textos están escritos en castellano y no existen traducciones a otros idiomas. Sería interesante añadir una traducción al inglés, como mínimo. Además, el código contiene una mezcla de castellano e inglés ya que las instrucciones propias del lenguaje de programación están en inglés, pero las variables y funciones están escritas en castellano por lo que puede llegar a ser confuso. La solución a este problema consiste en traducir las variables y funciones a inglés para que el código sea lingüísticamente coherente.

Las posibles mejoras que han sido descritas en este apartado son ampliaciones del trabajo realizado. Las bases del programa permiten que estas características sean desarrolladas en el futuro.

A continuación, se muestran los enlaces a la versión ejecutable y a los dos videotutoriales:

Versión ejecutable:

http://tiny.cc/DescargarProyecto o http://tiny.cc/DescargarProyecto2 (enlace alternativo).

Figura 44. Código QR 2. Elaboración propia.

Figura 45. Código QR 3. Elaboración propia.

Tutorial de instalación:

https://youtu.be/JtC1NjetLaQ

Figura 46. Código QR 4. Elaboración propia.

Tutorial de cómo usar el programa:

https://youtu.be/-e3rVOB_YXM

Figura 47. Código QR 5. Elaboración propia.

BIBLIOGRAFÍA

Arcotangente de dos parámetros. En Wikipedia. Recuperado el 5 de agosto de 2020 de https://es.wikipedia.org/wiki/Arcotangente_de_dos_par%C3%A1metros.

Esfera. Sin fecha. En Wikipedia. Recuperado el 13 de agosto de 2019 de https://es.wikipedia.org/wiki/Esfera#Coordenadas_sobre_la_esfera.

Lenguaje de programación. Sin fecha. En Wikipedia. Recuperado el 23 de julio de 2019 de https://es.wikipedia.org/wiki/Lenguaje_de_programación.

Plano (geometría). Sin fecha. En Wikipedia. Recuperado el 22 de julio de 2019 de https://es.wikipedia.org/wiki/Plano_(geometría).

Raffino, M. E. (6 de marzo de 2019). Dibujo técnico. Recuperado el 27 de junio de 2019, de https://concepto.de/dibujo-tecnico.

Raffino, M. E. (27 de noviembre de 2018). Lenguaje de programación. Recuperado el 4 de julio de 2019, de https://concepto.de/lenguaje-de-programacion.

BIBLIOGRAFÍA DE FIGURAS

Figura de la portada (captura de pantalla del programa): elaboración propia.

Logo de Python de la portada. Wikipedia (s.f.). *Python logo* [Figura]. Disponible en: https://es.wikipedia.org/wiki/Archivo:Python-logo-notext.svg

Logo de OpenGL de la portada. World Vector Logo (s.f.). *OpenGL vector logo* [Figura]. Disponible en: https://worldvectorlogo.com/es/logo/opengl-1

Logo de Qt de la portada. Pinclipart (s.f.). *Logo of the Qt Project* [Figura]. Disponible en: https://www.pinclipart.com/maxpin/hwxJmT/

Wikipedia (s.f.). *Coordenadas esféricas*. [Figura] Disponible en: https://es.wikipedia.org/wiki/Coordenadas_esf%C3%A9ricas#/media/Archivo:Coordenadas_esf%C3%A9ricas_figura.svg [Accedido el 6 Ene. 2020].