UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA
Escuela Profesional de Ingeniería de Sistemas

INFORME PROYECTO UNIDAD I

Curso: Diseño y Modelamiento Virtual

Docente: Ing. Hugo Manuel Barraza Vizcarra

TARQUI COLLATUPA, YERRY MAYCOL (2023078686)

Tacna – Perú
2025

INDICE

INFORME PROYECTO UNIDAD I

1. RESUMEN

El presente trabajo describe el desarrollo de un editor gráfico 2D utilizando OpenGL y GLUT, que permite dibujar líneas, círculos y elipses mediante algoritmos clásicos de rasterización, incluyendo línea Directa y DDA, círculo Incremental y Punto Medio, y elipse Punto Medio. El sistema cuenta con un menú interactivo para seleccionar tipo de figura, color, grosor, activar o desactivar cuadrícula y ejes, además de permitir la exportación de las figuras en formato PPM. También incorpora atajos de teclado para facilitar la interacción y mejorar la experiencia de usuario.

Se realizaron casos de prueba que comparan la precisión y comportamiento de los distintos algoritmos. Se observa que, aunque Incremental y Punto Medio pueden parecer similares visualmente, cada algoritmo utiliza fórmulas distintas, generando diferencias al analizar píxel a píxel. El informe finaliza con un análisis de ventajas y limitaciones de cada algoritmo, proporcionando una base sólida para futuras mejoras en la implementación gráfica y en la interfaz del editor.

Palabras clave: OpenGL, GLUT, rasterización, líneas, círculos, elipses, Punto Medio, Incremental, DDA, gráficos 2D, interacción.

2. INTRODUCCIÓN

El presente proyecto tiene como objetivo desarrollar un software CAD 2D básico en C++ utilizando FreeGLUT/OpenGL, que permite dibujar líneas, círculos y elipses mediante algoritmos clásicos de rasterización. El sistema incorpora un menú interactivo para seleccionar tipo de figura, algoritmo de dibujo, color, grosor y utilidades adicionales, así como una interfaz de interacción mediante mouse y teclado para capturar puntos, limpiar el lienzo, activar la cuadrícula y mostrar coordenadas.

Al finalizar este proyecto, el estudiante será capaz de configurar y compilar proyectos en C++ con FreeGLUT/OpenGL, implementar algoritmos de rasterización para líneas (Directo y DDA), círculos (Incremental y Punto Medio) y elipses (Punto Medio), manejar eventos de mouse y teclado, y validar los resultados mediante casos de prueba que contemplen diferentes pendientes, octantes, cuadrantes y radios o semiejes variados.

El sistema contempla una ventana de 800×600 píxeles con proyección ortográfica 2D, opción de cuadrícula y ejes visibles, y permite seleccionar puntos para dibujar cada figura según su tipo. Todos los trazos se realizan a nivel de píxel mediante GL_POINTS, sin utilizar primitivas completas de OpenGL, garantizando un control total del rasterizado. Se implementaron algoritmos específicos para cada figura, permitiendo analizar diferencias de precisión y visualización entre ellos.

Este proyecto proporciona una comprensión práctica de los algoritmos clásicos de rasterización, la interacción con gráficos 2D y la construcción de un sistema CAD básico. Además, establece la base para futuras mejoras, como la incorporación de antialiasing, nuevos algoritmos de dibujo y optimización de la interfaz gráfica.

3. OBJETIVOS

- Implementar algoritmos de rasterización para líneas: método Directo y DDA, considerando pendientes variables y casos especiales.

- Implementar algoritmos de rasterización para círculos: método Incremental y método del Punto Medio, utilizando simetría por octantes para optimizar los cálculos.

- Implementar el algoritmo de Punto Medio para el trazado de elipses, considerando las regiones 1 y 2 y la simetría por cuadrantes.

- Diseñar e integrar un menú interactivo que permita seleccionar figuras, algoritmos, color, grosor y utilidades como limpiar, deshacer y exportar imágenes en formato PPM.

- Validar la correctitud de los algoritmos mediante casos de prueba que analicen precisión visual, diferencias de cálculo y comportamiento en radios, semiejes y pendientes variadas.

4. MARCO TEÓRICO

4.1. Método Directo:

Este algoritmo utiliza la ecuación de la recta para calcular la coordenada **y** correspondiente a cada **x**. Se considera el valor absoluto de la pendiente **m** para decidir si se itera sobre **x** o **y**. El método requiere un redondeo al entero más cercano para ubicar los píxeles en la cuadrícula. Es simple de implementar, pero puede generar errores de acumulación en pendientes pronunciadas.

4.2. Método DDA:

El algoritmo DDA calcula los incrementos en **"x"** y **"y"** de manera fraccionaria y los acumula paso a paso, redondeando los resultados a enteros para ubicar los píxeles. Es más eficiente que el método directo en términos de consistencia de trazo y permite dibujar líneas con pendientes variables de forma uniforme.

4.3. Método Círculo Incremental:

El trazado de círculos incremental utiliza fórmulas basadas en trigonometría **(cos y sin)** para calcular los puntos de la circunferencia en cada paso, aprovechando la simetría de los octantes. Este método es intuitivo, pero puede introducir errores de redondeo al calcular las coordenadas de cada píxel.

4.4. Método Círculo Punto Medio:

Este algoritmo utiliza un parámetro de decisión **p** que determina si el siguiente píxel se coloca en la dirección horizontal o diagonal. Se aprovecha la simetría por octantes para minimizar cálculos. Es más eficiente y preciso que el método incremental, ya que evita operaciones trigonométricas y reduce errores de acumulación.

4.5. Método Elipse Punto Medio:

El algoritmo divide la elipse en dos regiones: la primera donde la pendiente es menor a 1 y la segunda donde es mayor. Se calcula un parámetro de decisión similar al de los círculos para determinar el siguiente píxel. Se utiliza la simetría por cuadrantes para dibujar toda la elipse a partir de los cálculos de un solo cuadrante. Este método garantiza un trazo uniforme y preciso sin depender de cálculos trigonométricos costosos.

5. DISEÑO DEL SISTEMA

5.1. Arquitectura General

El sistema CAD 2D se desarrolló en C++ usando Code::Blocks como entorno de desarrollo, con la biblioteca FreeGLUT/OpenGL para la interfaz gráfica. La arquitectura es monolítica, integrando todos los módulos principales dentro de la misma aplicación. Los componentes esenciales incluyen: interfaz gráfica, módulo de dibujo, manejo de eventos de mouse y teclado, menú interactivo y módulo de exportación de imágenes.

El flujo de funcionamiento es el siguiente: el usuario selecciona la figura y el algoritmo desde el menú, captura los puntos necesarios con el mouse (inicio y fin para líneas, centro y radio para círculos, centro y semiejes para elipses), y el algoritmo correspondiente genera los píxeles que conforman la figura. El usuario puede además activar la cuadrícula, los ejes o las coordenadas del puntero, y finalmente exportar la imagen en formato PPM.

Este diseño asegura un control total sobre el rasterizado, evitando el uso de primitivas completas de OpenGL, y facilita la futura extensión del sistema con nuevas figuras, algoritmos o utilidades.

5.2. Módulos

El sistema CAD 2D se estructura en módulos funcionales, cada uno con responsabilidades claras, implementados en C++ usando Code::Blocks y la biblioteca FreeGLUT/OpenGL. La separación modular permite mantener la organización del código y facilita la extensión futura del programa.

Módulo de gestión de figuras:

- Define la estructura Figura con los atributos alg (algoritmo), coordenadas x1, y1, x2, y2, color y grosor.
- Almacena todas las figuras dibujadas en el vector figuras, permitiendo su redibujo en cada actualización de la ventana.

Módulo de dibujo por algoritmo:

- Líneas: drawLineDirect implementa el método directo, y drawLineDDA implementa DDA.
- Círculos: drawCircleInc usa el método incremental, y drawCirclePM usa el método del punto medio.
- Elipses: drawEllipsePM implementa el método del punto medio considerando las regiones 1 y 2.
- Todos los métodos utilizan putPixel para controlar manualmente el rasterizado por píxeles.

Módulo de interfaz gráfica:

- display redibuja la escena incluyendo cuadrícula, ejes y figuras.
- drawGrid y drawAxes permiten mostrar u ocultar la cuadrícula y los ejes.
- drawText muestra coordenadas del puntero si se activa la opción correspondiente.
- Módulo de manejo de eventos:
- mouse y motion capturan las coordenadas del puntero y la selección de puntos de las figuras.
- keyboard gestiona teclas rápidas para activar/desactivar cuadrícula, ejes, limpiar pantalla o exportar imágenes.

Módulo de menú:

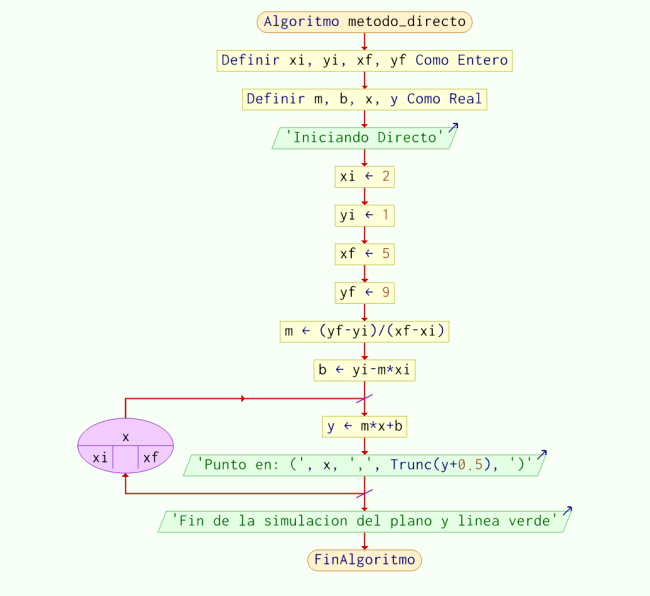
- createMenu define un menú contextual con submenús para:
- Dibujo (selección de algoritmo), Color, Grosor, Vista (cuadrícula, ejes, coordenadas), Herramientas (limpiar, borrar última figura, exportar), y Ayuda.
- Cada submenú llama a funciones específicas (menuAlg, menuColor, menuGrosor, etc.) para actualizar variables globales como algoritmoActual, colorActual y grosorActual.

Módulo de exportación:

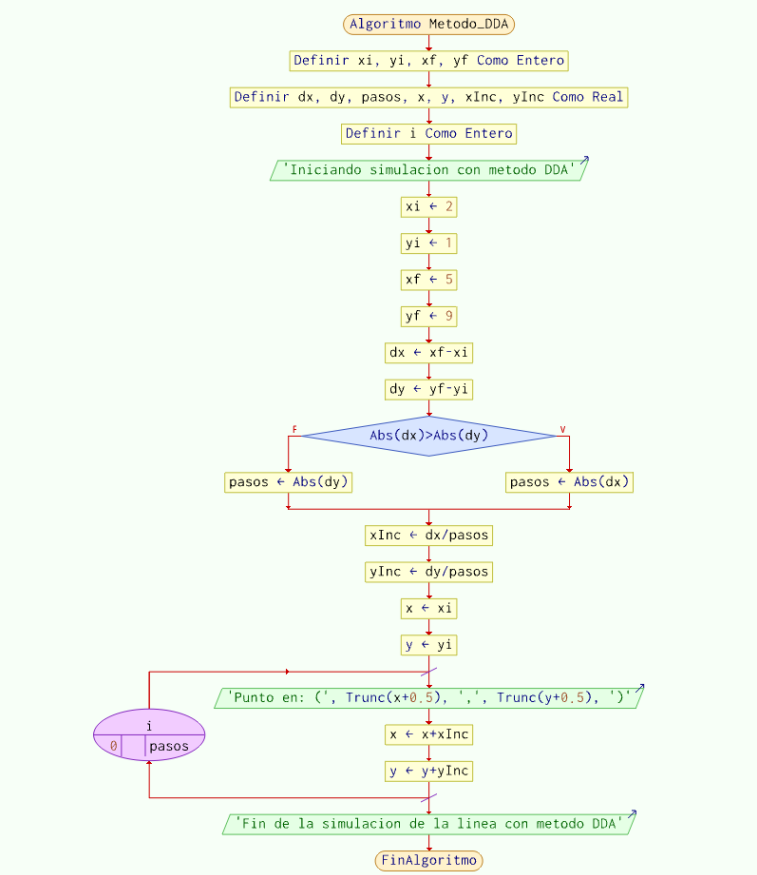
- exportToPPM captura los píxeles de la ventana OpenGL y genera archivos PPM para guardar la imagen.
- Mantiene un contador capturaCount para numerar automáticamente las imágenes exportadas.
- Cada módulo se comunica mediante variables globales (algoritmoActual, colorActual, grosorActual, etc.), garantizando que las acciones del usuario se reflejen correctamente en el dibujo y la actualización de la ventana.

5.3. Diagramas de flujo

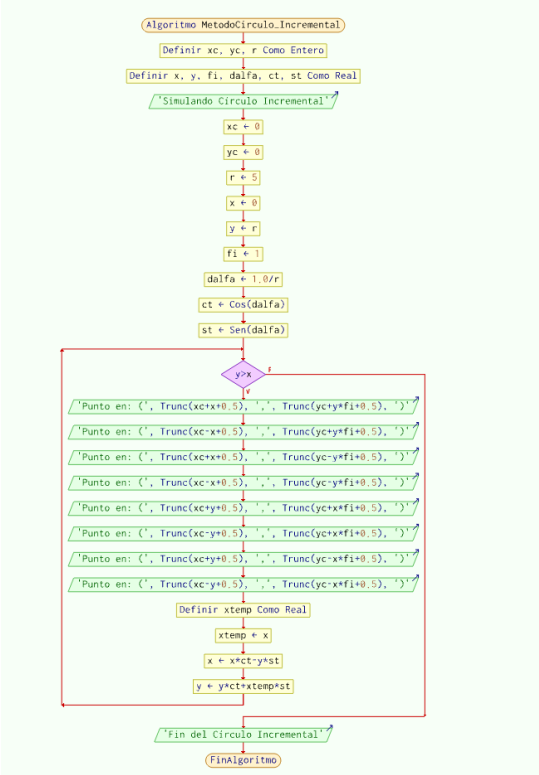
Método Directo



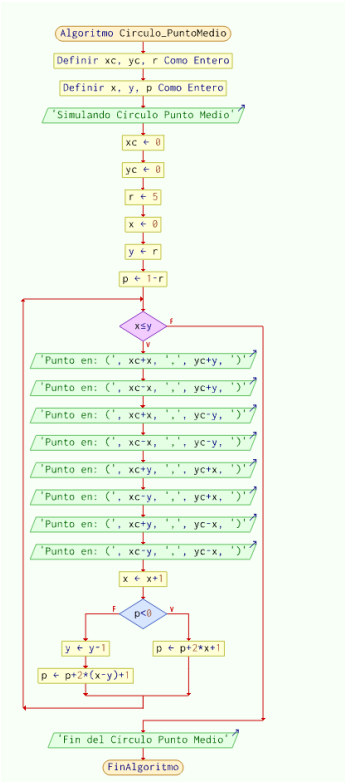
Método DDA



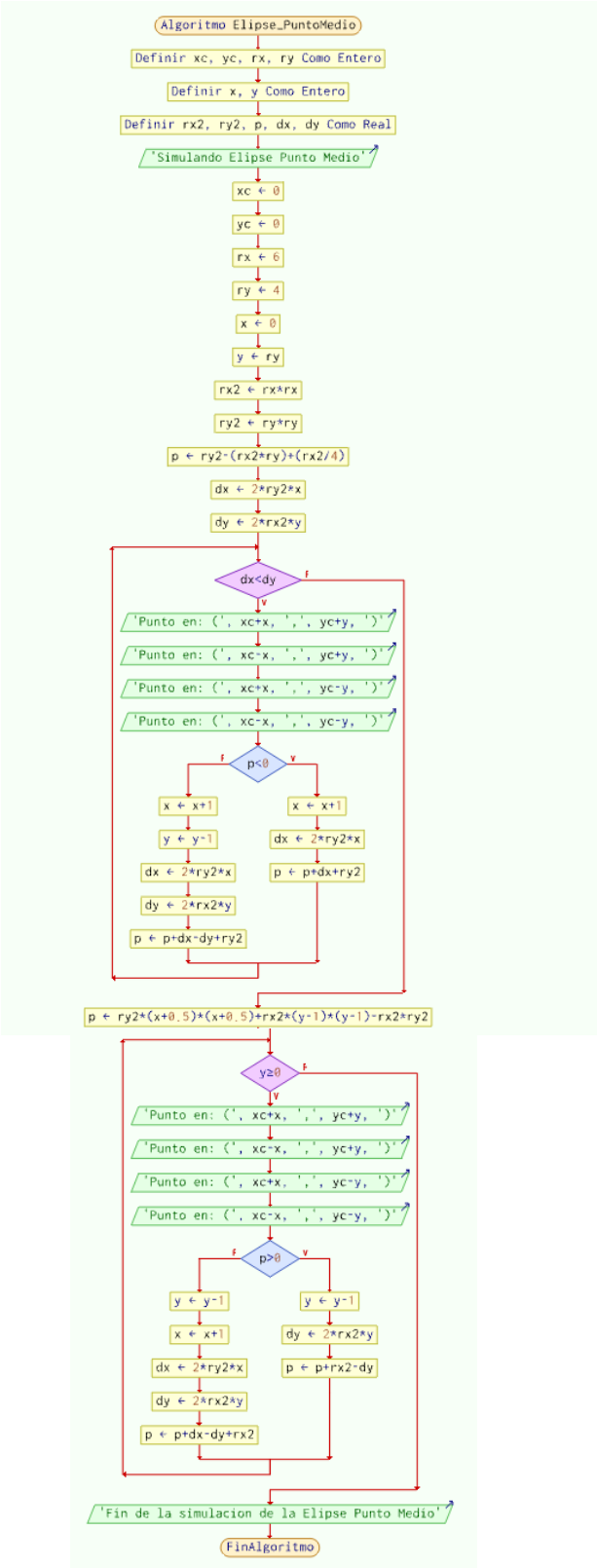
Método Círculo Incremental



Método Círculo Punto Medio



Método Elipse Punto Medio



6. ESPECIFACIÓN DEL MENÚ Y ATAJOS

MENU PRINCIPAL

Dibujo

- Recta Directa → selecciona el algoritmo de línea directa.
- Recta DDA → selecciona el algoritmo DDA.
- Círculo Incremental → selecciona el algoritmo de círculo incremental.
- Círculo Punto Medio → selecciona el algoritmo de círculo punto medio.
- Elipse Punto Medio → selecciona el algoritmo de elipse punto medio.

Color

- Negro
- Rojo
- Verde
- Azul

Grosor

- 1px, 2px, 3px, 5px

Vista

- Cuadrícula on/off
- Ejes on/off
- Coordenadas del puntero on/off

Herramientas

- Limpiar todas las figuras
- Borrar última figura
- Exportar PPM

Ayuda

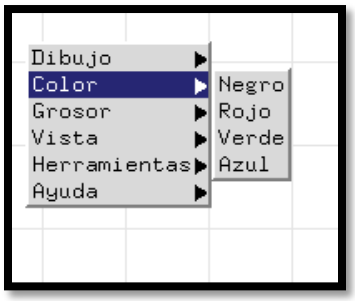
- Atajos
- Acerca de

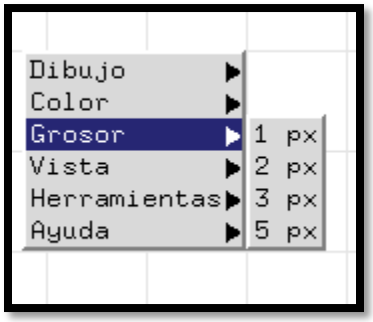
ATAJOS DEL TECLADO

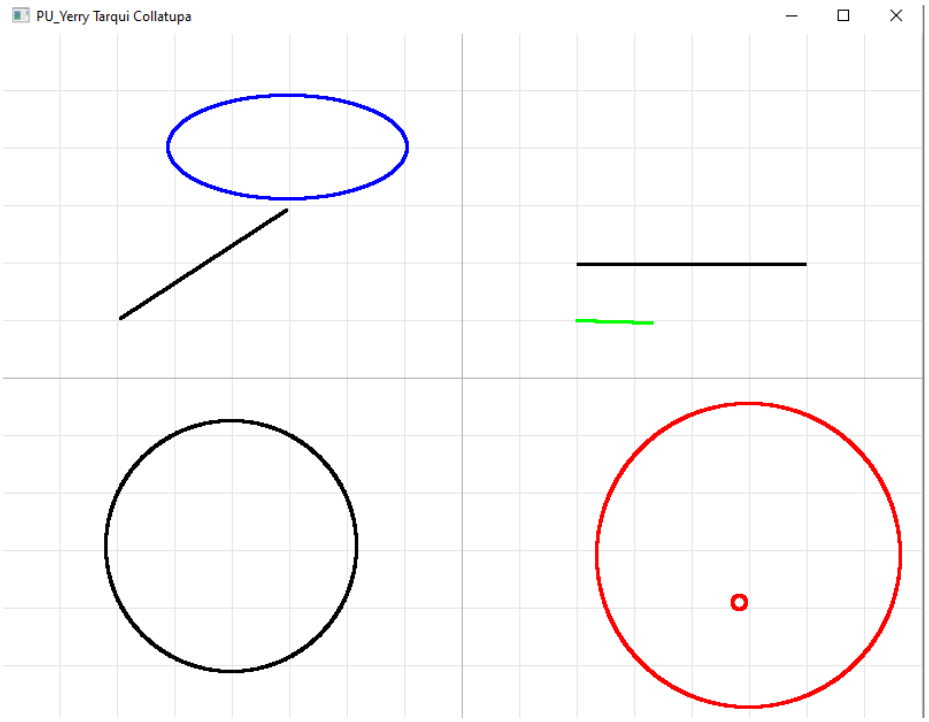
- G → mostrar/ocultar cuadrícula

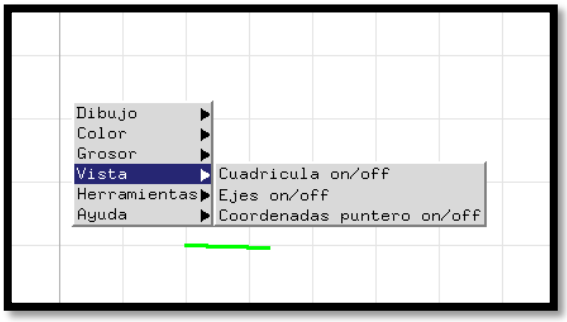
- E → mostrar/ocultar ejes
- C → limpiar todas las figuras
- S → exportar imagen

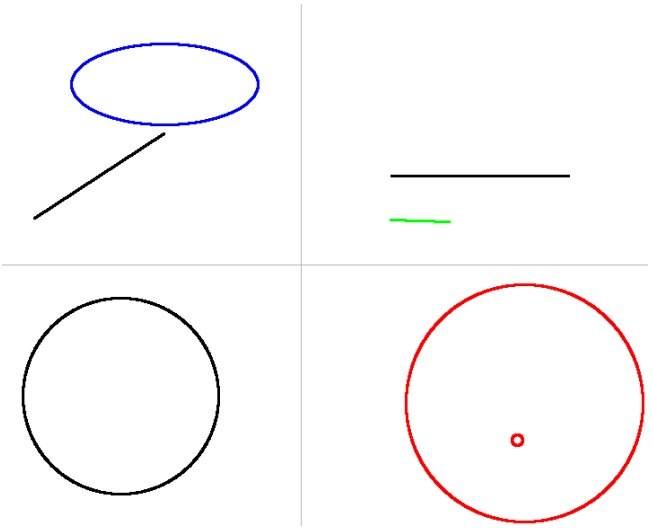
7. CASOS DE PRUEBA Y RESULTADOS

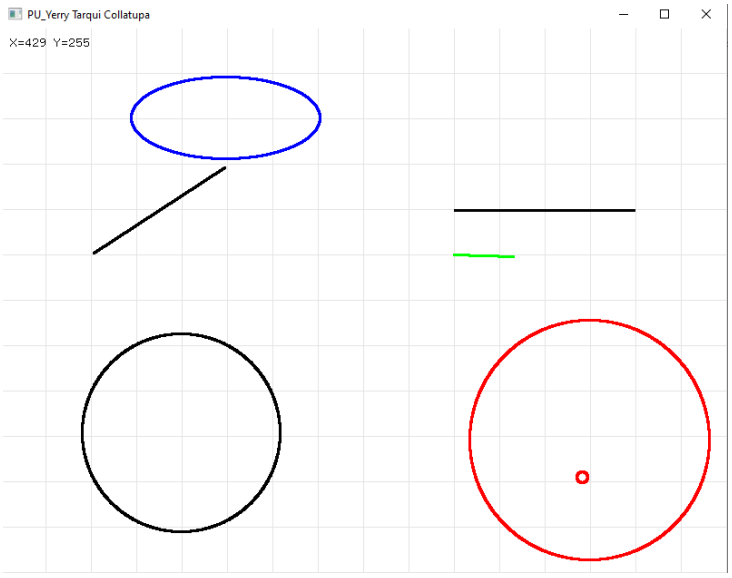


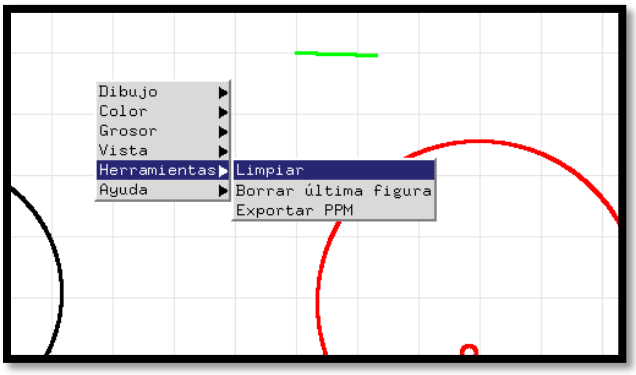


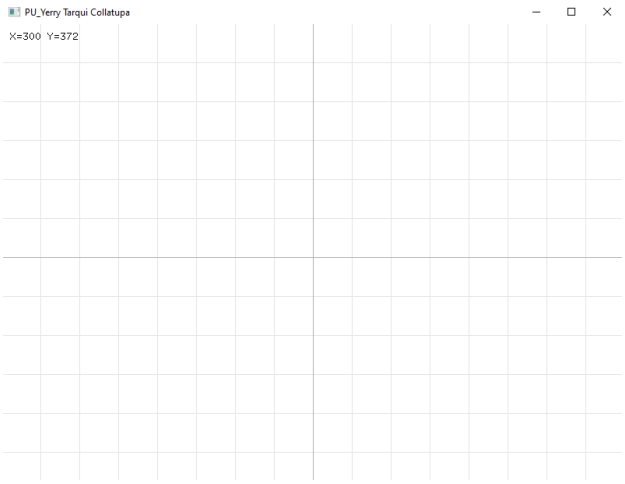



Dibujo
Color
Grosor
Vista
Herramientas
Ayuda
Cuadricula on/off
Ejes on/off
Coordenadas puntero on/off




PU_Yerry Tarqui Collatupa
X=429 Y=255


Dibujo
Color
Grosor
Vista
Herramientas
Ayuda
Limpiar
Borrar última figura
Exportar PPM


PU_Yerry Tarqui Collatupa
X=300 Y=372

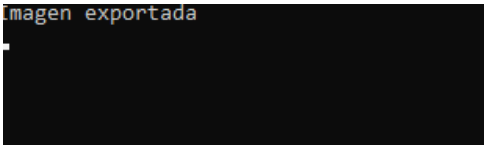
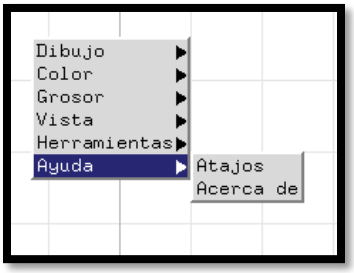
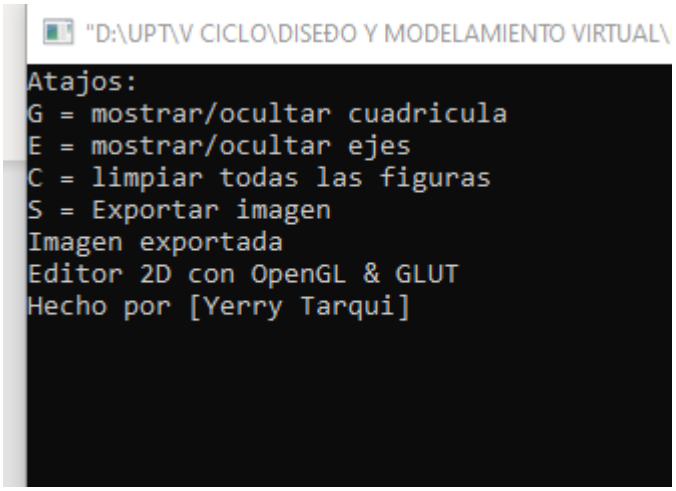


Imagen exportada


Dibujo
Color
Grosor
Vista
Herramientas
Ayuda
Atajos
Acerca de


"D:\UPT\V CICLO\DISEÐO Y MODELAMIENTO VIRTUAL\PU1_YERRY TA
Atajos:
G = mostrar/ocultar cuadricula
E = mostrar/ocultar ejes
C = limpiar todas las figuras
S = Exportar imagen
Imagen exportada


"D:\UPT\V CICLO\DISEÐO Y MODELAMIENTO VIRTUAL\
Atajos:
G = mostrar/ocultar cuadricula
E = mostrar/ocultar ejes
C = limpiar todas las figuras
S = Exportar imagen
Imagen exportada
Editor 2D con OpenGL & GLUT
Hecho por [Yerry Tarqui]

8. CONCLUSIONES

- Los algoritmos de línea (directo y DDA), círculo (incremental y punto medio) y elipse (punto medio) se implementaron correctamente y permiten calcular los puntos de cada figura de manera precisa.

- El trabajo con Code::Blocks permitió implementar y probar los algoritmos gráficos de manera práctica, demostrando cómo las funciones y procedimientos se traducen en trazos sobre el plano cartesiano, además de resaltar la importancia de un entorno de desarrollo integrado que simplifica la compilación, depuración y visualización de los resultados.

- Organizar el código en procesos y funciones, apoyado en diagramas de flujo, permitió entender la programación estructurada y planificar de manera ordenada la resolución de problemas gráficos.

- El desarrollo de este informe permitió integrar teoría y práctica sobre algoritmos de gráficos, comprendiendo cómo cada método de dibujo (línea directa, DDA, círculo incremental, círculo punto medio y elipse punto medio) funciona paso a paso, reforzando la importancia de planificar, simular y documentar los procesos antes de implementarlos en un entorno real de programación.

9. REFERENCIAS

Graficación. (2018, 9 febrero). Algoritmo de línea DDA. Recuperado de: https://2018graficacion.wordpress.com/2018/02/09/algoritmo-de-linea-dda/

Scribd. (s.f.). Algoritmo Bresenham y punto medio para elipses. Recuperado de: https://es.scribd.com/document/324905106/Algoritmo-Bresenham-y-punto-medio-para-Elipses