UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA
Escuela Profesional de Ingeniería de Sistemas

PROYECTO DE UNIDAD

Curso: DISEÑO Y MODELAMIENTO VIRTUAL

Docente: Ing. Refugio Valdivia Vargas

Concori Calizaya, Diego Anthony **(2023078689)**

Tacna – Perú
2025

INDICE

INTRODUCCION **3**
PROYECTO DE UNIDAD **4**
I. OBJETIVOS **4**
II. Marco teórico **5**
CONCLUSIONES **9**
BIBLIOGRAFIA **10**

INTRODUCCION

La **rasterización** es un proceso fundamental en gráficos por computadora que permite transformar representaciones matemáticas (ecuaciones y parámetros) en imágenes visibles sobre un dispositivo de salida, generalmente una pantalla. Su relevancia radica en que actúa como puente entre el mundo abstracto de la geometría analítica y el entorno tangible de los píxeles que conforman la interfaz visual de los usuarios.

A lo largo de la historia, diversos algoritmos han sido propuestos para resolver el problema de cómo representar rectas y curvas en dispositivos discretos. Entre los más influyentes destacan el **método directo**, el **algoritmo DDA (Digital Differential Analyzer)** y los algoritmos de **punto medio**, tanto para círculos como para elipses. Estos métodos constituyen la base de sistemas CAD (Computer-Aided Design) y de motores gráficos empleados en simulaciones, videojuegos y software de modelado.

El presente trabajo describe la implementación de un **Mini CAD 2D** desarrollado en C++ empleando la librería OpenGL/GLUT. El sistema permite al usuario interactuar mediante mouse, menús contextuales y atajos de teclado para crear figuras geométricas básicas. A través de este desarrollo se busca comprender no solo el funcionamiento de los algoritmos, sino también su impacto en la precisión, eficiencia y calidad visual de los gráficos generados.

En términos pedagógicos, este proyecto refuerza la comprensión de los conceptos fundamentales de los gráficos por computadora y ofrece una base sólida para abordar temas más avanzados como renderizado 3D, transformaciones geométricas y modelado.

PROYECTO DE UNIDAD

I. OBJETIVOS

1.1. GENERAL:

Desarrollar un sistema interactivo de representación de figuras geométricas básicas en 2D, empleando algoritmos clásicos de rasterización e implementado en C++ con OpenGL, integrando un menú contextual, cuadrícula y atajos de teclado.

1.2. ESPECIFICOS:

1.2.1. Implementar la rasterización de rectas mediante:

- Método Directo (ecuación de la recta).
- Algoritmo DDA.
- Comparación de exactitud y continuidad entre ambos métodos.

1.2.2. Desarrollar la rasterización de curvas:

- Círculo mediante el algoritmo de Punto Medio.
- Elipse mediante el algoritmo de Punto Medio dividido en dos regiones.

1.2.3. Diseñar la arquitectura modular del sistema, contemplando:

- Funciones de rasterización independientes.
- Módulo de gestión de menús y colores.
- Módulo de interacción mediante teclado y ratón.

1.2.4. Facilitar la interacción con el usuario mediante:

- Menú emergente con selección de figura y color.
- Atajos de teclado para mostrar/ocultar cuadrícula y ejes.
- Función para limpiar el lienzo.

1.2.5. Probar el sistema y documentar resultados:

- Capturas de rectas, círculos y elipses con diferentes configuraciones.
- Identificación de casos correctos y casos con errores intencionales.
- Discusión de limitaciones y rendimiento.

II. Marco teórico

- Método Directo (Ecuación y = mx + c): Calcula la pendiente y genera puntos uno a uno. Es sencillo, pero presenta problemas con pendientes muy grandes.
- DDA (Digital Differential Analyzer): Utiliza incrementos en x o y y es más preciso que el método directo. Requiere operaciones de punto flotante.
- Punto Medio para Círculo: Evalúa un parámetro de decisión para elegir entre dos píxeles vecinos, optimizando el cálculo y aprovechando la simetría en 8 octantes.
- Punto Medio para Elipse: Divide el cálculo en dos regiones para controlar el error de aproximación, generando una figura más precisa en los 4 cuadrantes.

III. Diseño del sistema

3.1. Arquitectura General

El Mini CAD 2D se construyó bajo una arquitectura modular, lo que permite separar las responsabilidades de cada componente y facilitar la comprensión del código. El sistema se organiza en:

- **Interfaz gráfica (OpenGL/GLUT):** Encargada del renderizado en la ventana, la gestión de eventos (ratón, teclado) y la interacción con menús.
- **Módulo de rasterización:** Contiene los algoritmos para representar figuras geométricas básicas en el plano discreto.
- **Módulo de control:** Gestiona el almacenamiento de las figuras, las banderas de visualización (cuadrícula y ejes) y los colores seleccionados.
- **Módulo de datos:** Estructura Figura y vector figuras para guardar los objetos que se dibujan.

3.2. Módulos del Sistema

3.2.1. Módulo de Rasterización

- **lineaDirecta:** Implementa la ecuación de la recta (y = mx + c).
- **lineaDDA:** Aplica el método incremental usando incrementos de punto flotante.
- **circulo:** Emplea el algoritmo del Punto Medio, optimizando con simetría en 8 octantes.
- **elipse:** Algoritmo del Punto Medio dividido en 2 regiones (pendiente < 1 y pendiente > 1).

3.2.2. Módulo de Visualización

- putPixel: Dibuja un punto en coordenadas (x,y).
- drawGrid: Renderiza cuadrícula configurable.
- drawAxes: Dibuja ejes cartesianos.
- display: Callback que redibuja todas las figuras almacenadas.

3.2.3. Módulo de Interacción

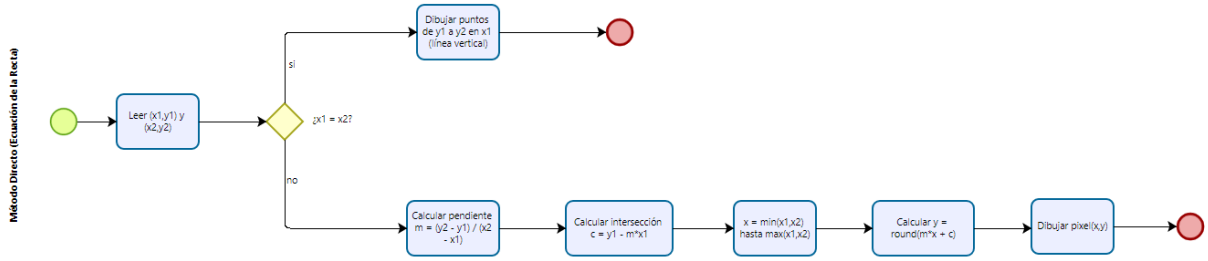
- **Mouse:** Captura coordenadas para definir figuras.
- **Teclado:** Atajos (g, e, c) para cuadrícula, ejes y limpiar lienzo.
- **Menú contextual:** Selección de figura y color con clic derecho.

3.2.4. Módulo de Datos

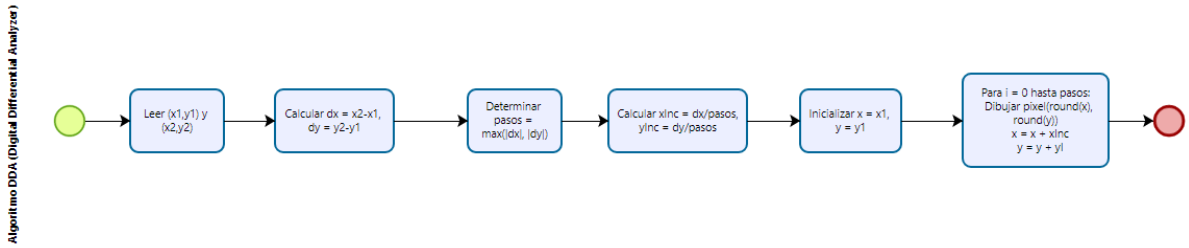
- **Estructura Figura:** Almacena tipo, puntos base y color.
- **Vector figuras:** Guarda todas las figuras creadas.

3.2.5. Diagramas de Flujo por Algoritmo

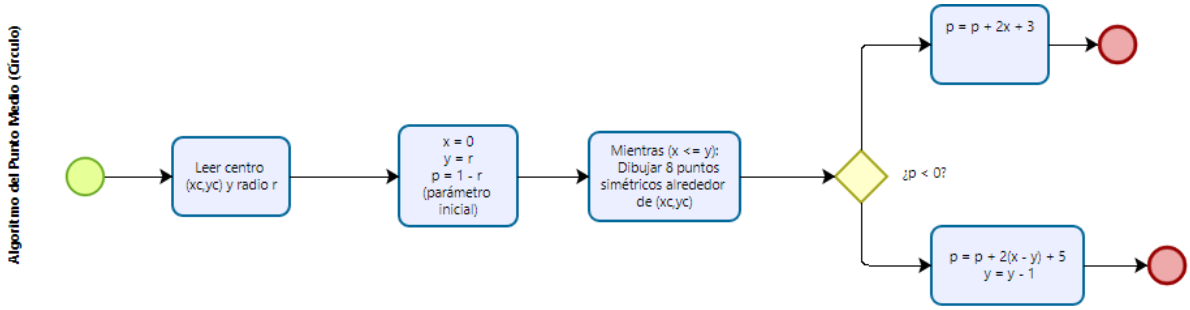
A. Línea por Método Directo



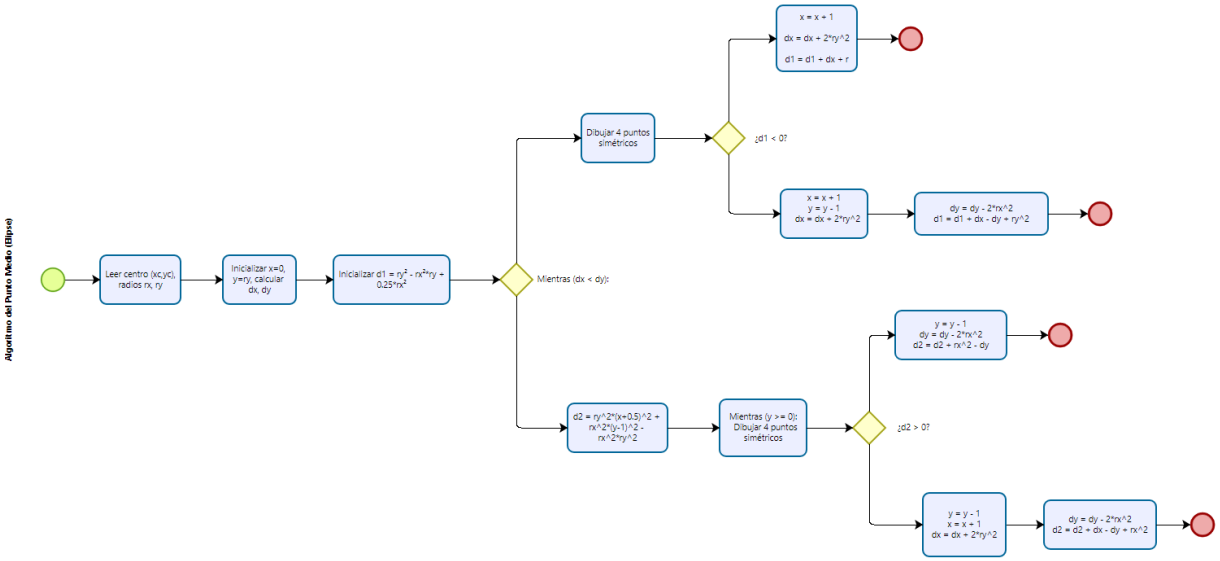
B. Línea por DDA (Digital Differential Analyzer)



C. Círculo por Algoritmo de Punto Medio



D. Elipse por Algoritmo de Punto Medio (2 regiones)



IV. Especificación del menú y atajos.

4.1. Menú emergente (clic derecho):

4.1.1. Figuras:

- Recta (Método Directo)
- Recta (Método DDA)
- Círculo (Punto Medio)
- Elipse (Punto Medio)

4.1.2. Colores:

- Negro
- Rojo
- Verde
- Verde

4.2. Atajos de teclado:

- g / G: Mostrar/ocultar cuadrícula.
- e / E: Mostrar/ocultar ejes.
- c / C: Limpiar el lienzo.

CONCLUSIONES

El desarrollo del **Mini CAD 2D** permitió implementar y comprender en profundidad los algoritmos clásicos de graficación de figuras geométricas básicas, como rectas, círculos y elipses. Se comprobó que los métodos implementados (DDA, Punto Medio, Método Directo) ofrecen soluciones eficientes y de bajo costo computacional para el dibujo en entornos gráficos.

Asimismo, se logró diseñar una arquitectura modular y una interfaz basada en menús y atajos de teclado, lo que facilita la interacción del usuario y la organización del código. La integración de cuadrícula y ejes cartesianos proporciona un entorno más amigable para el trazado de figuras, simulando las funcionalidades iniciales de un software CAD.

Trabajo Futuro

Como proyección, se identifican posibles mejoras al sistema:

- Implementar polígonos regulares e irregulares mediante coordenadas dinámicas.
- Añadir soporte para transformaciones geométricas (traslación, rotación, escalamiento).
- Incorporar herramientas de edición (mover, eliminar o modificar figuras).
- Guardar y cargar proyectos en formatos de imagen o archivos propios.
- Ampliar la gama de colores y estilos de línea.

Estas mejoras permitirían evolucionar el sistema hacia una aplicación más completa, cercana a un software CAD educativo.

BIBLIOGRAFIA

- Guha, S. (2022). *Computer Graphics through OpenGL: From Theory to Experiments* (4ª ed.). CRC Press.
- Gordon, V. S., & Clevenger, J. L. (2020). *Computer Graphics Programming in OpenGL with C++* (2ª ed.). Mercury Learning & Information.
- Scratchapixel. (s. f.). *An Overview of the Rasterization Algorithm*. Scratchapixel 2.0. https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm
- Microsoft Docs. (2020, 19 de agosto). *Rasterization Rules*. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/direct3d11/d3d10-graphics-programming-guide-rasterizer-stage-rules