UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA
Escuela Profesional de Ingeniería de Sistemas

PROYECTO DE UNIDAD I

Curso: Diseño y Modelamiento Virtual

Docente: Ing. HUGO MANUEL BARRAZA VIZCARRA

Cortez Apaza, Sebastian Alejandro (2023078699)

Tacna – Perú
2025

Introducción

El presente proyecto consiste en el diseño e implementación de un editor gráfico 2D denominado *Pixel CAD*, desarrollado en lenguaje C + + empleando las librerías OpenGL y GLUT. El sistema permite la creación y manipulación de primitivas geométricas básicas —rectas, círculos y elipses— a través de algoritmos de trazado ampliamente utilizados en la computación gráfica.

El propósito fundamental es comprender los principios de rasterización de figuras, mediante el uso de algoritmos clásicos como el método directo, el algoritmo incremental, el algoritmo digital diferencial analizador (DDA), y el algoritmo de punto medio tanto para círculos como para elipses. Estos procedimientos permiten representar gráficamente figuras sobre un lienzo digital mediante operaciones de bajo nivel, controlando el encendido de píxeles en pantalla.

Además, se busca integrar opciones de interacción con el usuario a través de menús contextuales y atajos de teclado, incorporando herramientas que simulan entornos CAD básicos, incluyendo cuadrícula, ejes, visualización de coordenadas, y exportación de resultados en formato de imagen.

El desarrollo del proyecto constituye una experiencia práctica que conecta la teoría algorítmica con la aplicación en tiempo real, reforzando el aprendizaje en el área de visualización y diseño de sistemas gráficos.

Objetivo general

- Implementar un editor gráfico 2D que permite trazar figuras geométricas básicas mediante algoritmos de rasterización clásicos, integrando menús, atajos de teclado y exportación de resultados.

Objetivos específicos

1. Implementar los algoritmos de trazado de rectas: método directo, incremental y DDA.

2. Implementar los algoritmos de punto medio para el trazado de círculos y elipses.

3. Diseñar un entorno gráfico con cuadrícula, ejes y coordenadas dinámicas para facilitar la construcción de figuras.

4. Incorporar un menú contextual y atajos de teclado para seleccionar algoritmos, colores, grosores y herramientas de edición.

5. Desarrollar la funcionalidad de exportación de la imagen generada en formato PPM.

6. Validar el funcionamiento del sistema mediante casos de prueba prácticos.

Marco teórico

Algoritmo directo para rectas

El algoritmo directo consiste en utilizar la ecuación de la recta en forma pendiente-intersección:

$$y=mx+by = mx + by=mx+b$$

donde mmm representa la pendiente y bbb la ordenada al origen. Para cada incremento en el eje xxx, se calcula el valor correspondiente en yyy. Si bien es sencillo, presenta problemas de precisión cuando la pendiente es elevada o no es entera, lo que genera saltos en la representación de los píxeles.

Algoritmo incremental para rectas

El método incremental optimiza el trazado de la recta realizando cálculos acumulativos a partir de una pendiente precomputada. Se reduce la cantidad de operaciones aritméticas repetitivas, mejorando la eficiencia respecto al método directo. Sin embargo, aún presenta problemas de redondeo acumulativo.

Algoritmo DDA (Digital Differential Analyzer)

El algoritmo DDA consiste en determinar el número de pasos en el eje con mayor desplazamiento (dxdxdx o dydydy) y calcular los incrementos fraccionarios en cada iteración. De esta forma, se genera una aproximación más uniforme de la recta en comparación con los métodos anteriores, siendo uno de los algoritmos más utilizados en la rasterización inicial de gráficos por computadora.

Algoritmo de punto medio para circunferencia

El algoritmo de punto medio aprovecha la simetría de la circunferencia para calcular únicamente un octante, replicando los resultados en los demás. A partir de un parámetro de decisión, se determina si el siguiente punto debe trazarse hacia el este o hacia el sureste, reduciendo la complejidad computacional y evitando cálculos con raíces cuadradas.

Algoritmo de punto medio para elipse

De manera análoga al círculo, el algoritmo de punto medio para la elipse evalúa un parámetro de decisión en dos regiones: la parte superior (pendiente menor a 1) y la parte inferior (pendiente mayor a 1). Con ello, se logran representar las elipses de manera eficiente y precisa, utilizando únicamente operaciones aritméticas enteras.

Diseño del sistema

Arquitectura general

El sistema está compuesto por los siguientes módulos principales:

1. **Interfaz gráfica**: ventana de dibujo, cuadrícula, ejes y coordenadas dinámicas.

2. **Módulo de algoritmos**: implementación de rectas (directo, incremental, DDA), circunferencia y elipse por punto medio.

3. **Módulo de interacción**: manejo de eventos de teclado, ratón, menús contextuales y atajos.

4. **Módulo de exportación**: captura de pantalla y almacenamiento en formato PPM.

Diagramas de flujo (por algoritmo)

- **Recta directa**: inicio → leer puntos extremos → calcular pendiente mmm → iterar sobre xxx o yyy → graficar píxeles.

- **Recta incremental**: inicio → leer puntos extremos → calcular incremento acumulativo → actualizar posición → graficar píxeles.

- **DDA**: inicio → calcular dx, dy → determinar pasos → incrementar x e y proporcionalmente → graficar píxeles.

- **Circunferencia (punto medio)**: inició → radio rrr → parámetro de decisión → iterar octante → reflejar en simetrías → graficar píxeles.

- **Elipse (punto medio)**: inicio → radios rx, ry → evaluar región 1 y 2 → parámetro de decisión → graficar píxeles con simetrías.

Especificación del menú y atajos

Menú contextual (clic derecho)

- **Dibujo**: Recta Directa, Recta Incremental, Recta DDA, Círculo PM, Elipse PM.
- **Color**: Negro, Rojo, Verde, Azul, Amarillo (personalizado).
- **Grosor**: 1 px, 2 px, 3 px, 5 px.
- **Vista**: Cuadrícula on/off, Ejes on/off, Coordenadas on/off.
- **Herramientas**: Limpiar, Borrar última figura, Exportar PPM.
- **Ayuda**: Atajos, Acerca de.

Atajos de teclado

- **G**: mostrar/ocultar cuadrícula.
- **E**: mostrar/ocultar ejes.
- **C**: limpiar lienzo.
- **S**: exportar imagen.
- **Z**: deshacer.
- **Y**: rehacer.

Casos de prueba y resultados

Se realizaron pruebas dibujando rectas, círculos y elipses en diferentes posiciones y tamaños, utilizando los distintos algoritmos implementados.

- **Caso 1**: recta horizontal, vertical y diagonal con cada algoritmo → resultados coinciden con la teoría.
- **Caso 2**: círculo de radio grande y pequeño → simetría adecuada y trazo uniforme.
- **Caso 3**: elipse con radios diferentes → representación correcta en ambos ejes.
- **Caso 4**: cambio de color (rojo, verde, azul, amarillo) y grosor (1–5 px) → resultados visibles en pantalla.
- **Caso 5**: uso de atajos (G, E, C, S, Z, Y) → funcionalidades correctas en tiempo real.
- **Caso 6**: exportación en PPM → imagen generada en la carpeta de ejecución.

Capturas de pantalla sugeridas:

- Vista general con cuadrícula y ejes.
- Figura con zoom resaltando el trazado de píxeles.
- Ejemplo de recta, círculo y elipse exportados.

Conclusiones

- Los algoritmos de trazado implementados funcionan correctamente y permiten comprender el fundamento de la rasterización en gráficos por computadora.

- La integración de un menú contextual y atajos de teclado facilita la interacción con el usuario, asemejando el entorno a un software CAD básico.

- La cuadrícula, los ejes y la visualización de coordenadas mejoran la precisión en el diseño.

- La funcionalidad de exportación asegura la persistencia de los resultados generados.

- Como trabajo futuro, se plantea implementar transformaciones geométricas (traslación, rotación, escalado) y soporte para formatos de imagen más comunes como PNG o BMP.

Referencias

- Wikipedia contributors. (2025, 13 agosto). *OpenGL*. Wikipedia. https://en.wikipedia.org/wiki/OpenGL
- colaboradores de Wikipedia. (2024, 27 enero). *GLUT*. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/GLUT