DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

COMPUTACIÓN GRÁFICA

Implementación de Algoritmos Gráficos Básicos

Nombre: Kerly Chuqui

NRC: 23463

Fecha: 16/06/2025

Tema:

Implementación de Algoritmos Gráficos Básicos

Objetivo General

Desarrollar una aplicación gráfica en C# utilizando Windows Forms que implemente algoritmos gráficos básicos, permitiendo el trazado y visualización animada de líneas, circunferencias y polígonos, así como el relleno de figuras, aplicando conceptos de programación orientada a objetos y una interfaz amigable para el usuario.

Objetivos Específicos

1. Implementar los algoritmos DDA y Bresenham para el trazado de líneas entre dos puntos seleccionados por el usuario.
2. Desarrollar el algoritmo de Bresenham para circunferencias con entrada dinámica del centro y el radio.
3. Aplicar el algoritmo de relleno por inundación para figuras geométricas como polígonos regulares y circunferencias, mejorando la interacción con herramientas gráficas.

Marco Teórico

1. Algoritmo DDA (Digital Differential Analyzer)

El algoritmo DDA es un método de interpolación digital que genera puntos entre dos coordenadas para simular una línea recta. Calcula los incrementos en x y y y avanza en pasos hasta completar el trazado. Es preciso pero más lento que otros métodos porque trabaja con números flotantes.

Ventajas: Precisión.

Desventajas: Uso de float, menor rendimiento comparado con Bresenham.

2. Algoritmo de Bresenham para Líneas

Bresenham es un algoritmo eficiente de trazado de líneas basado en operaciones enteras. Evalúa el error de aproximación y decide si debe aumentar x, y o ambos. Por esta razón, es ampliamente usado en renderizado a bajo nivel.

Aplicación: Dibujar líneas en dispositivos rasterizados (pantallas).

3. Algoritmo de Bresenham para Circunferencias

Extiende la lógica del Bresenham para trazar circunferencias usando simetría octogonal. Calcula puntos en un octante y refleja a los otros siete.

Uso: Visualización de curvas cerradas como esferas o bordes de objetos.

4. Relleno por Inundación (Flood Fill)

Este algoritmo colorea una región conectada alrededor de un punto dado, expandiéndose en 4 o 8 direcciones hasta alcanzar un borde o un color diferente. Se utiliza una pila o cola para controlar los puntos pendientes de evaluar.

Desventajas: Requiere control del color objetivo y puede ser lento si no se optimiza.

Construcción del Proyecto

- **Lenguaje y entorno:** C# en Visual Studio 2022 con Windows Forms.
- **Estructura modular:** Se dividió el código en clases independientes por tipo de algoritmo (AlgoritmosLinea, AlgoritmoCircunferencia, AlgoritmoRelleno, etc.).
- **Interfaz amigable:** Botones para cada algoritmo, PictureBox para el lienzo, ListBox para mostrar coordenadas encendidas y ColorDialog para selección de color.
- **Interacción usuario:** El usuario selecciona puntos directamente con el mouse o define valores como el número de lados de un polígono.
- **Animación progresiva:** Implementada con async/await para ver encenderse los píxeles uno por uno.

Análisis del Funcionamiento

- **DDA y Bresenham:** Se activan después de que el usuario selecciona dos puntos (inicio y fin). El trazado es animado pixel a pixel.
- **Circunferencia:** El usuario define el centro y luego el radio, generando el trazado circular.
- **Polígono:** El usuario ingresa el número de lados. Se calcula con trigonometría y se unen vértices con Bresenham.
- **Relleno:** El programa detecta el centro de la figura como punto de partida y aplica el algoritmo Flood Fill si el área está cerrada.

Recomendaciones

1. Validar siempre los rangos de los puntos antes de pintar en el Bitmap para evitar errores de índice o argumentos inválidos.
2. Usar un lienzo (canvas) separado y redibujarlo en pantalla con Refresh() para evitar parpadeo o errores gráficos.
3. Implementar una forma de almacenar las figuras dibujadas para volver a rellenarlas sin necesidad de redibujar manualmente.

Conclusiones

- Se logró implementar exitosamente algoritmos gráficos clásicos en un entorno moderno (C# con Windows Forms).
- El uso de programación orientada a objetos permitió mantener una estructura clara y escalable del proyecto.
- La interacción directa del usuario con el dibujo de figuras, selección de color y animación visual, mejora la comprensión de los algoritmos subyacentes y su funcionamiento paso a paso.

Enlace al Git Hub:

https://github.com/kerly2001andreina/DeberesCGrafi