UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Asignatura: Computación Gráfica – NRC28467

Deber - Unidad 2
Tema: Algoritmos Clásicos de la Computación Gráfica

Nombre:

Báez, Gabriel

Profesor: Ing. Dario Javier Morales Caiza

Fecha: 8 de diciembre de 2025

Sangolquí – Ecuador

Índice

1. Introducción

El repositorio implementa un conjunto de algoritmos clásicos de computación gráfica: trazado de líneas y círculos, relleno de regiones y operaciones de *clipping* tanto para líneas como para polígonos. Las implementaciones están orientadas a la claridad pedagógica y a la robustez: validan entradas, manejan casos degenerados y documentan supuestos (por ejemplo, convexidad en ciertas rutinas).

Este informe lista cada algoritmo, su implementación en el proyecto, los parámetros principales, justifica por qué se eligió la variante implementada y compara opciones alternativas indicando casos de uso recomendados.

2. Algoritmos de trazado de líneas

2.1. CDDAAlgorithm

Archivo: `CDDAAlgorithm.cs`
Método principal:

```
public List<Point> GenerateLinePoints(int x0, int y0, int x1, int y1)
```

Descripción técnica: El algoritmo DDA calcula incrementos

$$xInc = \frac{dx}{steps}, \qquad yInc = \frac{dy}{steps},$$

y genera la línea interpolando con redondeo mediante `Math.Round`.

Ventajas: simple, flexible y apropiado para interpolación fina. **Desventajas:** usa aritmética de punto flotante.

2.2. CBresenhamAlgorithm

Archivo: `CBresenhamAlgorithm.cs`
Descripción técnica: Implementación clásica entera con término de error:

$$err = dx - dy$$

Ventajas: extremadamente rápido y preciso sin coma flotante. **Desventajas:** más complejo que DDA conceptualmente.

2.3. CMidPointAlgorithm

Archivo: `CMidPointAlgorithm.cs`
Algoritmo basado en el punto medio, equivalente en rendimiento a Bresenham pero pedagógico para comprender la evaluación incremental de decisiones.

3. Algoritmos de círculo

3.1. CParametricCircleAlgorithm

Archivo: `CParametricCircleAlgorithm.cs`
Utiliza la forma paramétrica:

$$x = x_c + r\cos\theta, \qquad y = y_c + r\sin\theta$$

Permite generar círculos y arcos con control del muestreo.

3.2. CCircle (Midpoint Circle)

Archivo: `CCircle.cs`
Implementación entera del algoritmo del punto medio, usando simetría de 8 octantes.

4. Algoritmos de recorte (clipping) de líneas

4.1. CCohenSutherland

Archivo: `CCohenSutherland.cs`
Método basado en *outcodes* para clasificar regiones y aplicar pruebas triviales.

4.2. CLiangBarsky

Archivo: `CLiangBarsky.cs`
Método paramétrico más eficiente (O(1)) que Cohen-Sutherland.

4.3. CCyrusBeck

Generalización a polígonos convexos mediante normales externas y parámetros t_E, t_L.

5. Algoritmos de recorte de polígonos

5.1. CSutherlandHodgman

Funciona por etapas, recortando el polígono entrada contra cada arista del polígono recortador.

5.2. CWeilerAtherton

Permite recorte con polígonos cóncavos y múltiples regiones resultantes.

5.3. CGreinerHormann

Versión robusta moderna que maneja auto-intersecciones y casos degenerados.

6. Algoritmos de relleno

6.1. CFloodFillAlgorithm

Relleno BFS en 4-conectividad, usando `GetPixel` y `SetPixel`.

6.2. CBoundaryFill

Incluye tres variantes:

- **FourConnected**
- **EightConnected**
- **OptimizedEightConnected**

6.3. CScanlineFill

Implementación con Active Edge Table (AET) compatible con reglas de relleno Even-Odd y NonZero.

7. Parámetros comunes y constantes

Se usa una tolerancia numérica

$$\varepsilon = 10^{-6}$$

para comparaciones robustas.

Los métodos validan:

- polígonos con al menos 3 vértices,
- rectángulos con área positiva,
- convexidad cuando se requiere.

8. Tablas comparativas

Las siguientes tablas provienen de tu documento original. Se presentan alineadas a la página, sin desbordes y utilizando formato profesional `booktabs`.

8.1. Recorte de Líneas

Criterio	**Cohen-Sutherland**	**Liang-Barsky**	**Cyrus-Beck**
Año / Autor	1968	1984	1978
Tipo de ventana	Rectángulo alineado	Rectángulo alineado	Polígono convexo
Complejidad temporal	O(1)	O(1)	O(n)
Operaciones principales	Outcodes, intersecciones iterativas	Tests paramétricos (t0,t1)	Normales y dot-products
Robustez numérica	Alta	Media	Alta
Facilidad de implementación	Muy fácil	Fácil	Moderada
Uso recomendado	Educación / muchas líneas fuera	Producción raster	Polígonos convexos

Cuadro 1: Características de algoritmos de recorte de líneas

8.2. Recorte de Polígonos

Criterio	**Sutherland-Hodgman**	**Weiler-Atherton**	**Greiner-Hormann**
Requisito clip convexo	Sí	No	No
Soporta sujeto cóncavo	Sí	Sí	Sí
Múltiples polígonos resultantes	No	Sí	Sí
Complejidad	O(n·m)	O(n·m + k)	O(n·m + k)
Robustez	Media	Alta	Muy alta
Facilidad implementación	Sencillo	Complejo	Complejo
Uso recomendado	Clip convexo	Clip cóncavo	Casos complejos

Cuadro 2: Comparación de algoritmos de recorte de polígonos

8.3. Trazado de Líneas y Círculos

Criterio	**DDA**	**Bresenham**	**MidPoint (línea)**	**Parametric Circle**	**Midpoint Circle**
Tipo	Flotante	Entero	Entero	Paramétrico	Entero simétrico
Complejidad	O(steps)	O(steps)	O(steps)	O(n)	O(r)
Precisión	Buena	Exacta	Exacta	Depende del muestreo	Exacta
Ventaja	Control de muestreo	Rápido	Pedagógico	Arcos flexibles	Muy eficiente
Desventaja	Coste flotante	Sin AA	Similar	Trigonométrico	No arcos directos

Cuadro 3: Comparación de métodos de trazado

8.4. Métodos de Relleno

Algoritmo	Tipo	Complejidad	Ventaja	Recomendación
Flood Fill (BFS)	4-conectividad	O(area)	Simple	Regiones pequeñas
Boundary Fill	Borde detenido	O(area)	Control frontera	Variante scanline
Scanline Fill	AET	O(h + edges)	Muy eficiente	Polígonos grandes

Cuadro 4: Métodos de relleno

8.5. Rendimiento Estimado

Caso	Mejor opción	Observación
Línea dentro del rectángulo	Cohen-Sutherland	Aceptación trivial
Línea que cruza	Liang-Barsky	Rápido y estable
Segmento vs polígono convexo	Cyrus-Beck	Maneja aristas arbitrarias
Polígono con muchas intersecciones	Greiner-Hormann	Mejor que S-H
Relleno de polígonos grandes	Scanline	Eficiencia alta

Cuadro 5: Rendimiento estimado por caso

8.6. Validación de Inputs

Algoritmo	Validaciones
Cohen-Sutherland	Rectángulo válido
Liang-Barsky	Rectángulo y no degenerados
Cyrus-Beck	Polígono convexo
Sutherland-Hodgman	3 vértices, convexo
Weiler-Atherton	Limpieza de vértices
Flood/Boundary/Scanline	Coordenadas válidas

Cuadro 6: Validaciones de entrada típicas

8.7. Optimización

Optimizaciones recomendadas
• Migrar SetPixel a LockBits
• Vectorizar clipping masivo
• Wu's algorithm para antialias
• Índices espaciales para polígonos grandes

Cuadro 7: Recomendaciones de optimización

8.8. Complejidad Cognitiva

Algoritmo	**Nivel**	**Notas**
Sutherland-Hodgman	Baja	Fácil de verificar
Cohen-Sutherland	Baja	Didáctico
Liang-Barsky	Baja	Paramétrico claro
Cyrus-Beck	Media	Requiere normales
Weiler-Atherton	Alta	Traversal complejo
Greiner-Hormann	Alta	Robustez alta
Scanline Fill	Media	Manejo de AET

Cuadro 8: Complejidad de implementación

8.9. Puntuación

Objetivo	**LB**	**Cohen**	**S-H**	**G-H**	**Scanline**
Velocidad	5	4	5	3	5
Robustez	4	4	3	5	4
Facilidad	4	5	5	2	3
Generalidad	3	2	2	5	4

Cuadro 9: Puntuación comparativa (1–5)

9. Análisis comparativo

Líneas

- **Bresenham**: mejor rendimiento y precisión.
- **Midpoint**: educativo y muy similar en rendimiento.
- **DDA**: preferible para interpolaciones suaves o antialiasing.

Círculos

- **Midpoint Circle**: ideal para círculos completos.
- **Paramétrico**: recomendado para arcos.

Clipping de líneas

- **Liang-Barsky**: más eficiente.
- **Cohen-Sutherland**: intuitivo y bueno con rechazo rápido.

10. Limitaciones y mejoras sugeridas

- Optimizar `GetPixel`/`SetPixel` usando `LockBits`.
- Manejo más robusto de polígonos degenerados.
- Implementación paralela para rellenos de gran escala.

11. Pruebas y validación recomendadas

- Verificación de octantes en líneas.
- Pruebas con polígonos:
  - convexos,
  - cóncavos,
  - auto-intersectados.
- Comparación visual de clipping y relleno.

12. Conclusiones

El proyecto implementa un conjunto robusto, pedagógico y completo de algoritmos fundamentales en computación gráfica. La combinación de variantes otorga flexibilidad, claridad conceptual y comparabilidad directa entre métodos clásicos e implementaciones modernas.

13. Recomendaciones

- Usar Liang-Barsky en entornos productivos.
- Usar Bresenham para trazado general de líneas.
- Usar Scanline para relleno eficiente de polígonos.