Introducción a la Computación Gráfica

Ing. Gabriel Ávila, MSc.

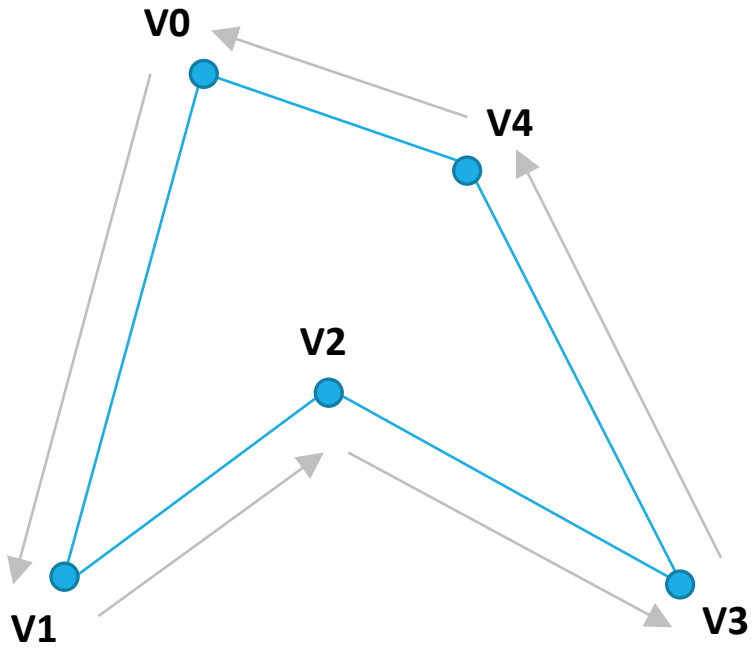
Conjunto ordenado de vértices, usualmente en sentido contrario a las manecillas del reloj.

Dos vértices consecutivos conforman un **borde** o **arista**.

A la izquierda del vértice se encuentra la parte interna del polígono, a su derecha la parte externa.

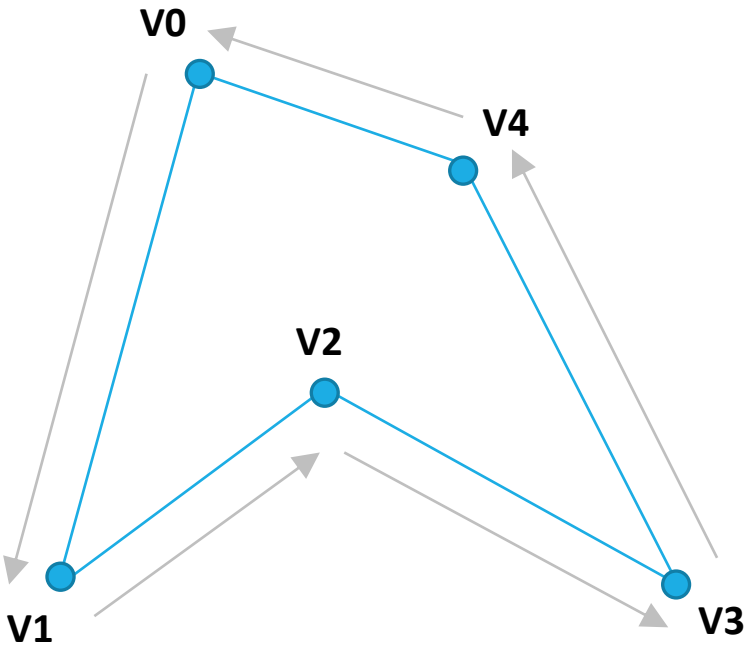
El último vértice está conectado con el primero.

En 3D, los vértices son coplanares.



Polígonos

¿Qué píxeles se deben rellenar?
¿Con qué valores se deben rellenar?



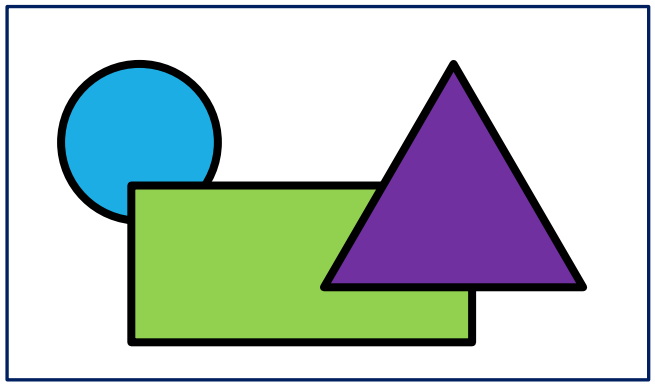
Relleno de figuras

POLYGON FILLING

Relleno de figuras

Proceso:

1. Dibujar la primitiva, a partir de los algoritmos de Bressenham, DDA o punto medio.
2. Rellenar la primitiva usando:
   - Algoritmo de líneas de barrido (scan line).
   - Algoritmo de relleno por frontera (boundary-fill).
   - Algoritmo de relleno por difusión (flood-fill).

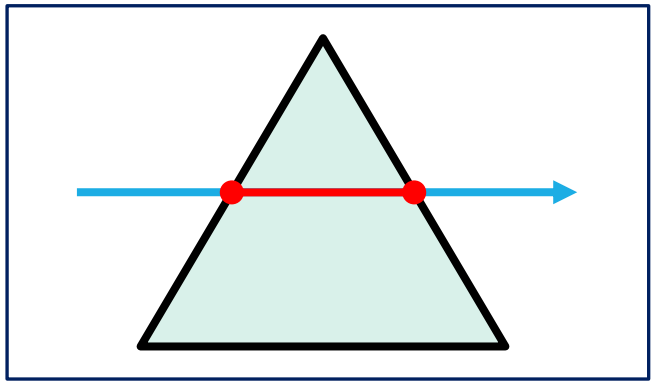


Definición:

Línea que recorre la pantalla manteniendo y constante, avanzando pasos unitarios en x.

Es necesario identificar los puntos de intersección de la línea de barrido con la región a rellenar. Estos puntos son los que definen la primitiva.

Se pueden almacenar en una lista ordenada con respecto a la coordenada *y*.

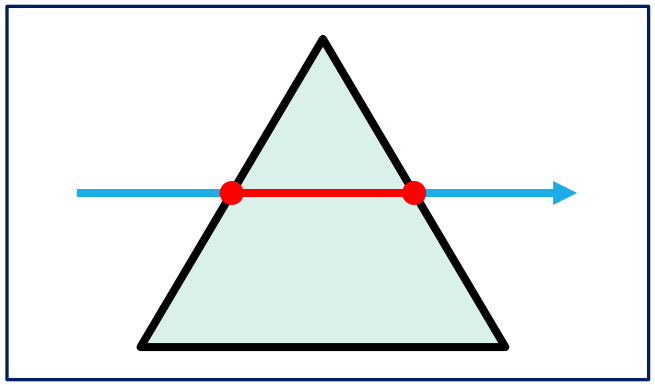


Líneas de barrido – *Scan lines*

Algoritmo de líneas de barrido

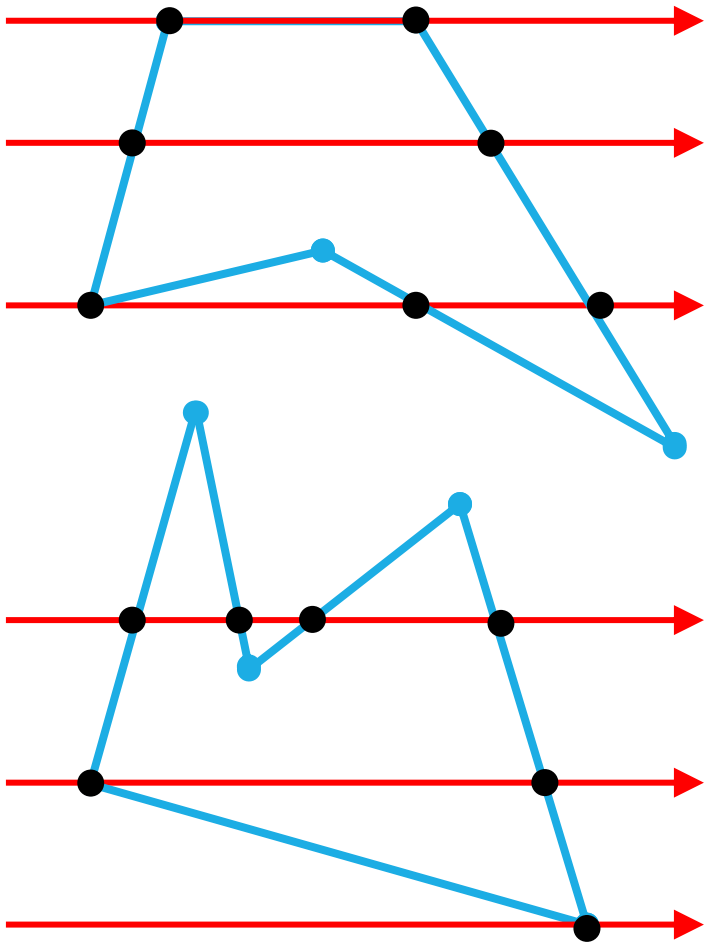
Para cada línea de barrido:

1. Encuentre las intersecciones de la línea con los bordes del polígono.
2. Ordene las intersecciones en orden ascendente en x.
3. Rellene todos los pixeles entre cada par de intersecciones, usando una variable de paridad B_P
   - B_P inicia en false.
   - En cada intersección se invierte el valor de B_P.
   - Rellenar cuando B_P sea true.



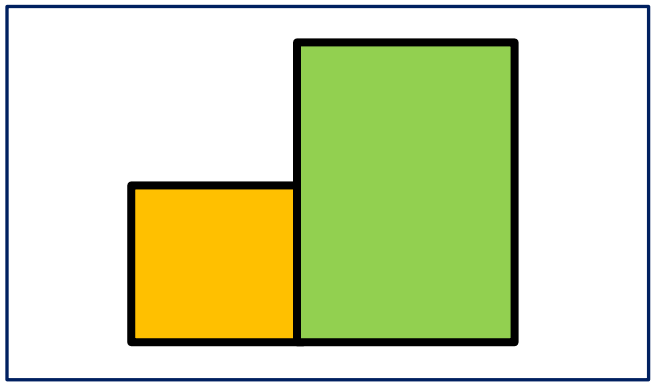
Es necesario verificar los puntos de intersección:

- **Número par**, se rellena entre cada par de puntos.
- **Si es un vértice**, contarlo como doble y no rellenar. ¿Funciona siempre?
- **Si es una línea horizontal**, no tener en cuenta sus vértices. ¿Cómo saber si es una línea horizontal?
- Sólo rellenar los pixeles cuyos centros estén dentro del polígono (Redondeo selectivo).



Líneas de barrido

Líneas de barrido – Rectángulos



Algoritmo sencillo:

1. Rellenar desde x_{min} hasta x_{max}.
2. Rellenas desde y_{min} hasta y_{max}.

Inconvenientes:

¿Qué hacer si dos rectángulos comparten un borde? ¿Colorear los bordes dos veces?

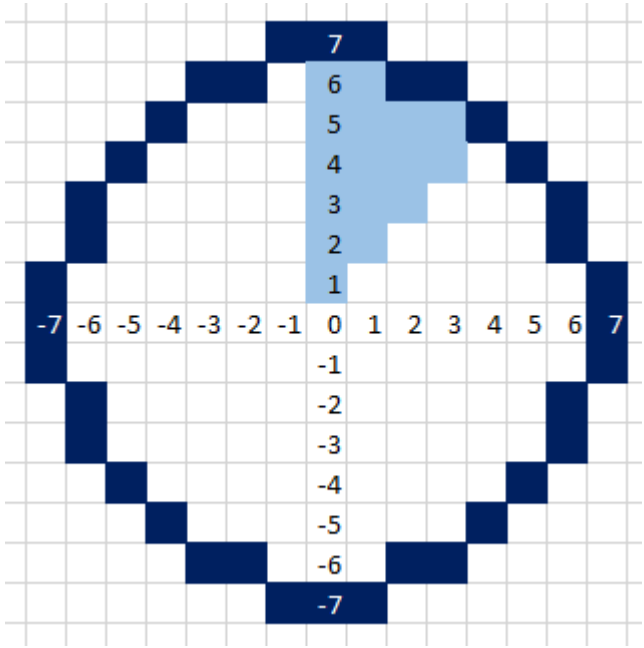
Reglas:

- Colorear solo los pixeles internos.
- Colorear los bordes izquierdo e inferior del rectángulo.

Para curvas en general puede ser más complejo.

Para círculos y elipses:

1. Calcular el borde con el algoritmo de punto medio.
2. Realizar el barrido en el segundo octante.
3. Los demás octantes por simetría.

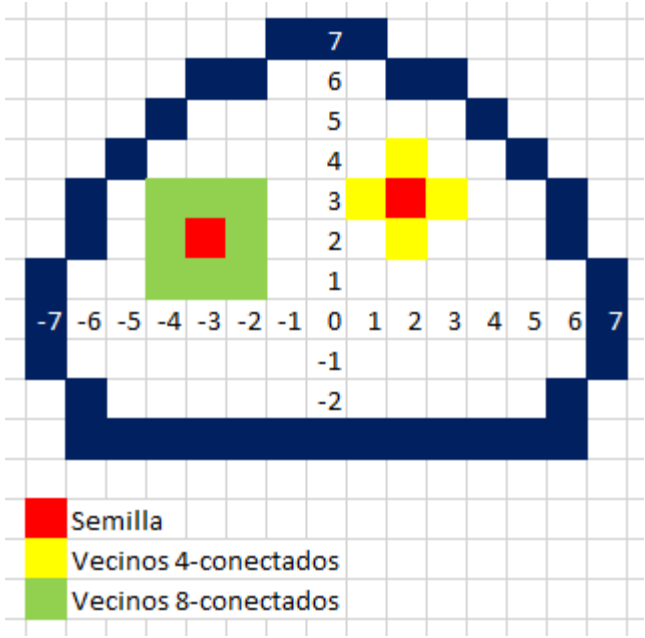


Líneas de barrido – Curvas

Consiste en recorrer el interior de un polígono, buscando los bordes y rellenando en las diferentes direcciones.

1. Seleccionar un punto dentro del polígono (semilla).
2. Colorear dicho punto.
3. Verificar si los vecinos del punto tienen color de relleno o de borde. Pueden ser los vecinos *4-conectados* u *8-conectados*.
4. Colorear los pixeles vecinos que no estén coloreados.
5. Continuar de manera recursiva hasta colorear toda la figura.

¿Qué pasa si hay zonas que ya tienen el color de relleno?



Por frontera – *Boundary fill*

```
void BoundaryFill4( int x, int y, int fill, int border) {
  if ( Color(x,y) != fill && Color(x,y) != border) {
    SetColor(x,y) = fill;
    BoundaryFill4( x+1,   y, fill, border );
    BoundaryFill4(   x, y+1, fill, border );
    BoundaryFill4( x-1,   y, fill, border );
    BoundaryFill4(   x, y-1, fill, border );
  }
}
```

¡¡Al ser recursivo, consume muchos recursos del PC (stack)!!

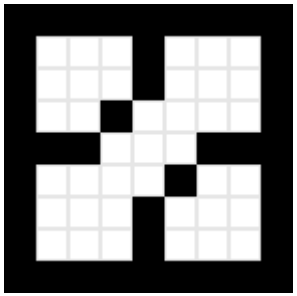
Solución: *Primero pintar a lo largo de una línea de barrido. Agregar a la pila las posiciones iniciales de nuevas líneas de escaneo.*

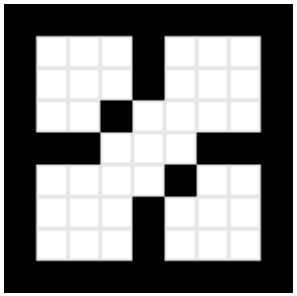
Adaptado de [6]

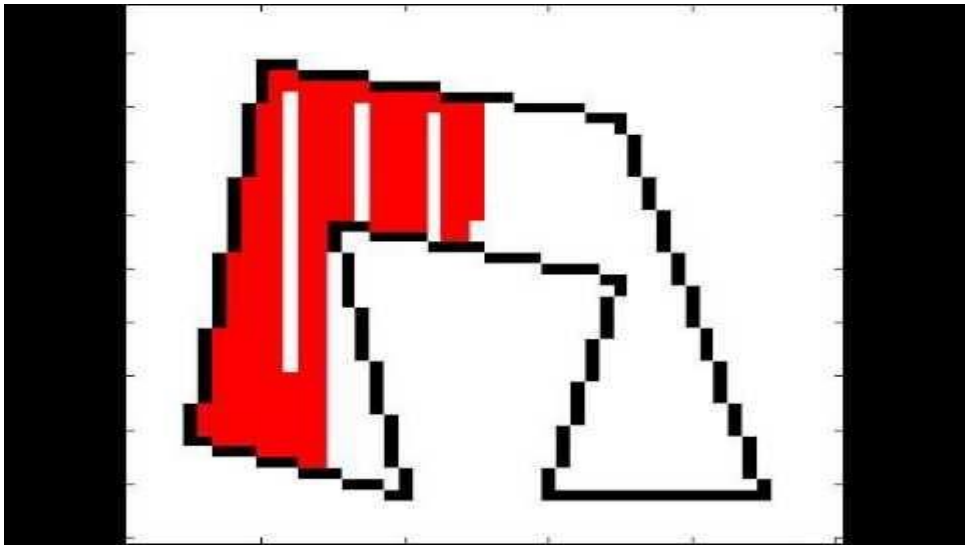
Relleno por frontera - Código

Similar al relleno por frontera.

No se tiene en cuenta el borde, solo el color del interior, mientras que haya pixeles del color original, se sigue dibujando.







Relleno por difusión – *Flood fill*

En grupos de proyecto, escoger e implementar alguno de los algoritmos vistos:

1. Implementar el algoritmo de líneas de barrido, para rellenar un polígono de n vértices (usando la clase línea propia).
2. Implementar el algoritmo de relleno por frontera.
3. Implementar el algoritmo de relleno por difusión.

Ejercicio en clase

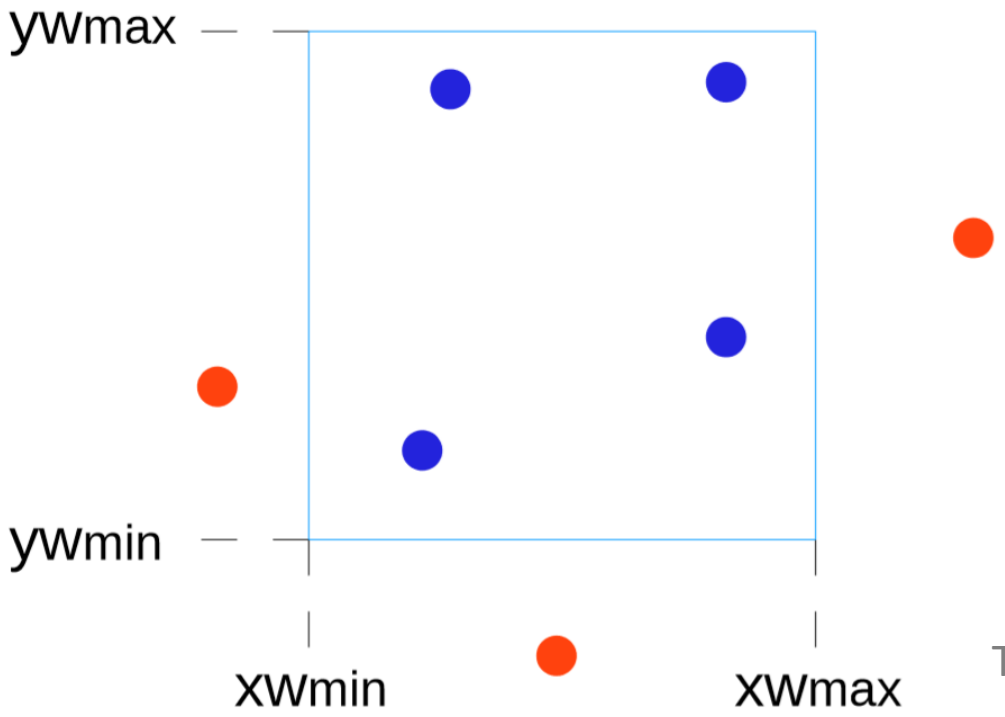
Recorte de figuras

POLYGON CLIPPING

Se dibuja el punto P(x,y) si está dentro de la ventana de dibujo, es decir:

$$xw_{min} \leqslant x \leqslant xw_{max}$$

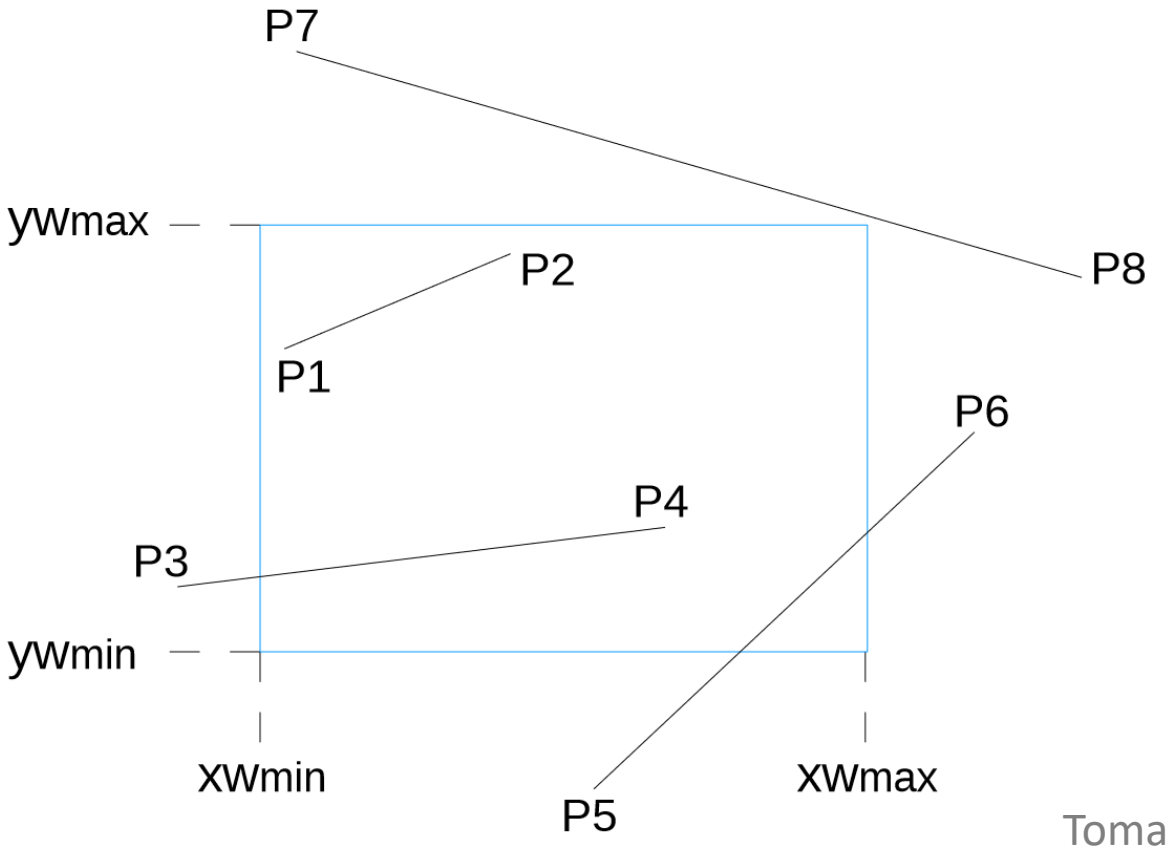
$$yw_{min} \leqslant y \leqslant yw_{max}$$



Tomado de [1]

Recorte de puntos

Un poco más complejo!!!



Tomado de [1]

Recorte de líneas

Recorte de líneas

Diversos métodos, basados en la ubicación de los vértices de la línea y su corte con los extremos de la ventana

- ***Cohen-Sutherland***
- ***Liang-Barsky***
- ***Nicholl-Lee-Nicholl***

Hacen uso de la representación paramétrica de la línea:

$$x = x_0 + u(x_f - x_0)$$

$$y = y_0 + u(y_f - y_0)$$

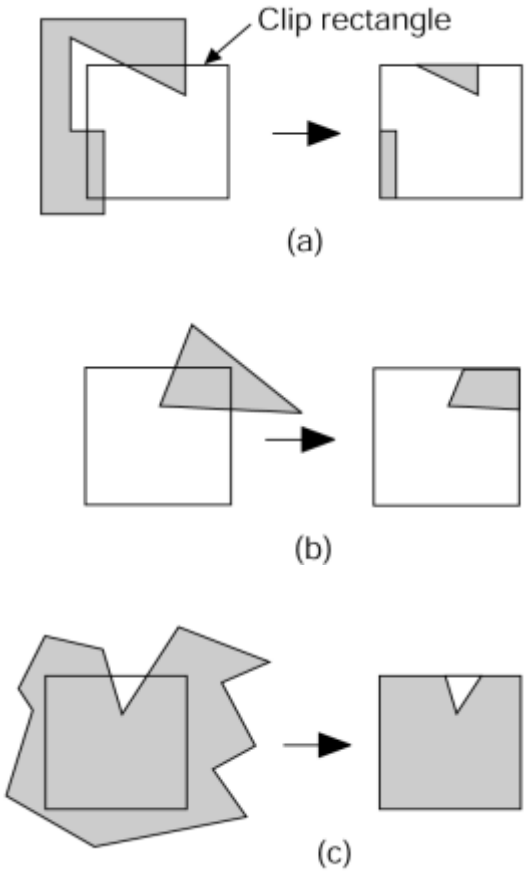
$$\boxed{0 \leqslant u \leqslant 1}$$

Recorte de polígonos

Los bordes de cada polígono requieren ser probados con respecto a un rectángulo de recorte.

Existen múltiples casos:

- Puede ser necesario agregar nuevos bordes.
- Pueden surgir varios polígonos a partir de uno solo.
- Algunos bordes se descartan o se dividen.



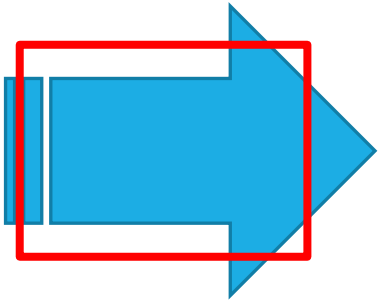
Tomado de [6]

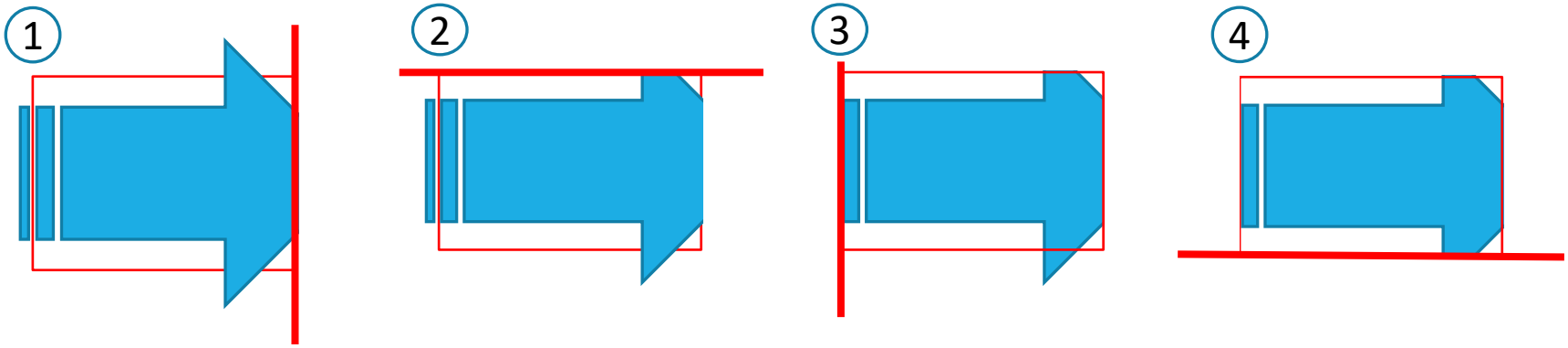
Divide y vencerás.

Se realiza el recorte utilizando uno de los bordes del rectángulo de recorte, prolongándolo de manera infinita.

Se repite para los 4 bordes.

¿Qué pasa con polígonos cóncavos?





Algoritmo de Sutherland-Hodgman

Datos de entrada:

- Los vértices que definen el polígono: $v_1, v_2, \dots\ v_n$
- Borde de recorte infinito, con la información de interior y exterior.

Datos de salida:

- Los vértices del polígono recortado: $v_1', v_2', \dots v_m'$

ALGORITMO:

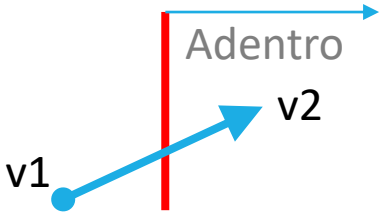
Repetir 4 veces (para una ventana de recorte rectangular):

1. Recorrer entre pares de vértices (arista del polígono).
2. Añadir vértices, uno a la vez, al polígono de salida.
   - Usando la información de interior/exterior.
   - Las intersecciones de las aristas.

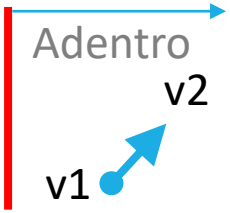
Algoritmo de Sutherland-Hodgman

Existen 4 casos diferentes de relación, entre cada arista del polígono y el borde de recorte.

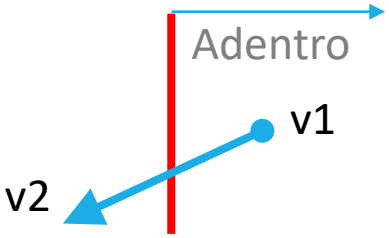
Caso 1: v_1 fuera del borde y v_2 dentro del borde.



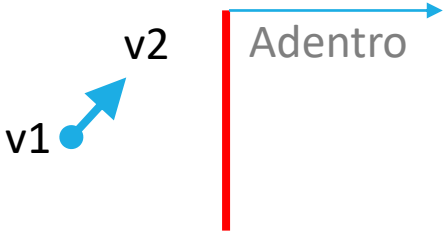
Caso 2: v_1 y v_2 dentro del borde de la ventana.



Caso 3: v_1 dentro del borde y v_2 fuera del borde.



Caso 4: v_1 y v_2 fuera del borde de la ventana.

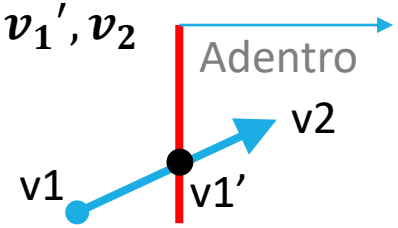


Algoritmo de Sutherland-Hodgman

Existen 4 casos diferentes de relación, entre cada arista del polígono y el borde de recorte.

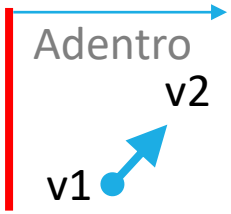
Caso 1: v_1 fuera del borde y v_2 dentro del borde.

Salida: $\boldsymbol{v_1', v_2}$



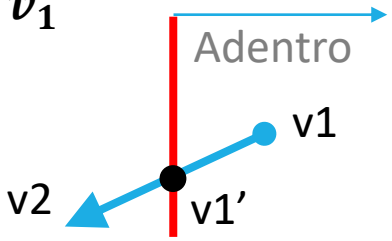
Caso 2: v_1 y v_2 dentro del borde de la ventana.

Salida: $\boldsymbol{v_2}$



Caso 3: v_1 dentro del borde y v_2 fuera del borde.

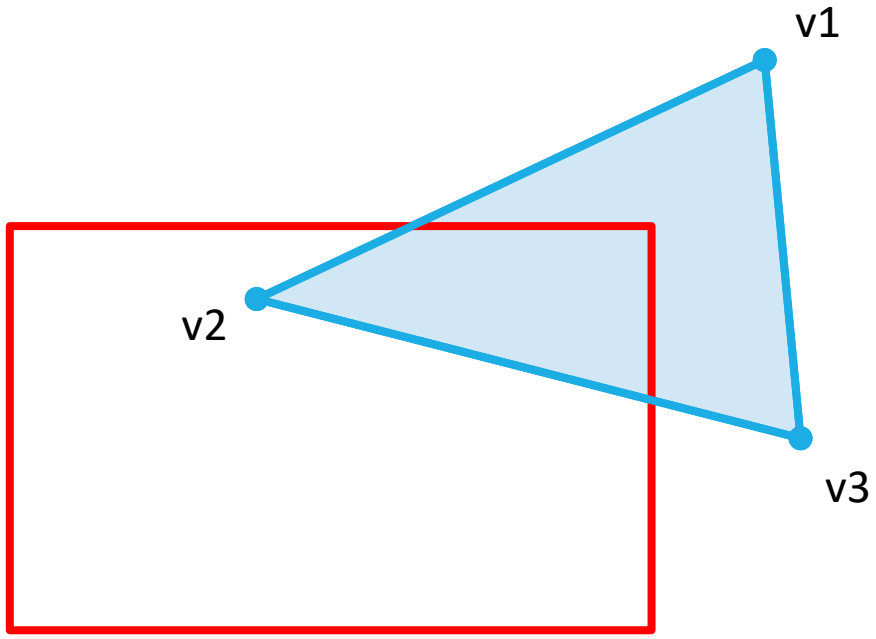
Salida: $\boldsymbol{v_1'}$



Caso 4: v_1 y v_2 fuera del borde de la ventana.

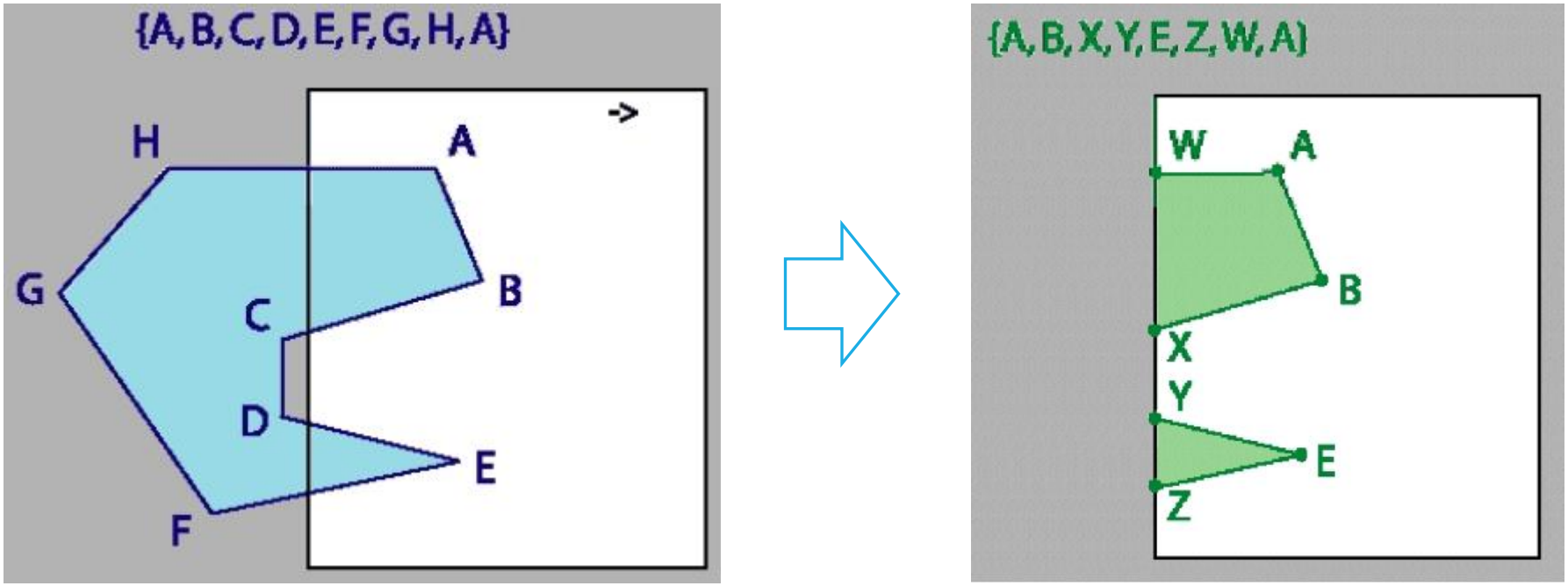
Salida: ninguna

Algoritmo de Sutherland-Hodgman



Ejercicio

Se generan aristas entre WZ y XY



Tomado de [6]

Problema – Polígonos cóncavos

Tiene en cuenta polígonos cóncavos, con huecos, generando múltiples regiones de relleno (polígonos independientes).

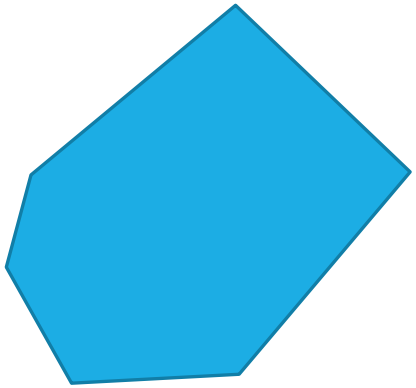
Algoritmo:

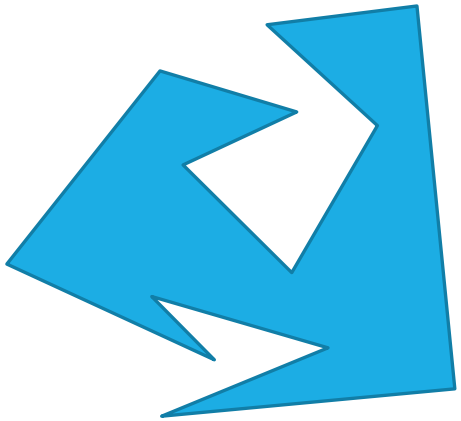
Dados dos polígonos P1 y P2, representados como listas ordenadas de vértices (en sentido contrario a las manecillas),

Weller-Atherton

Trabajar con triángulos es mas sencillo:

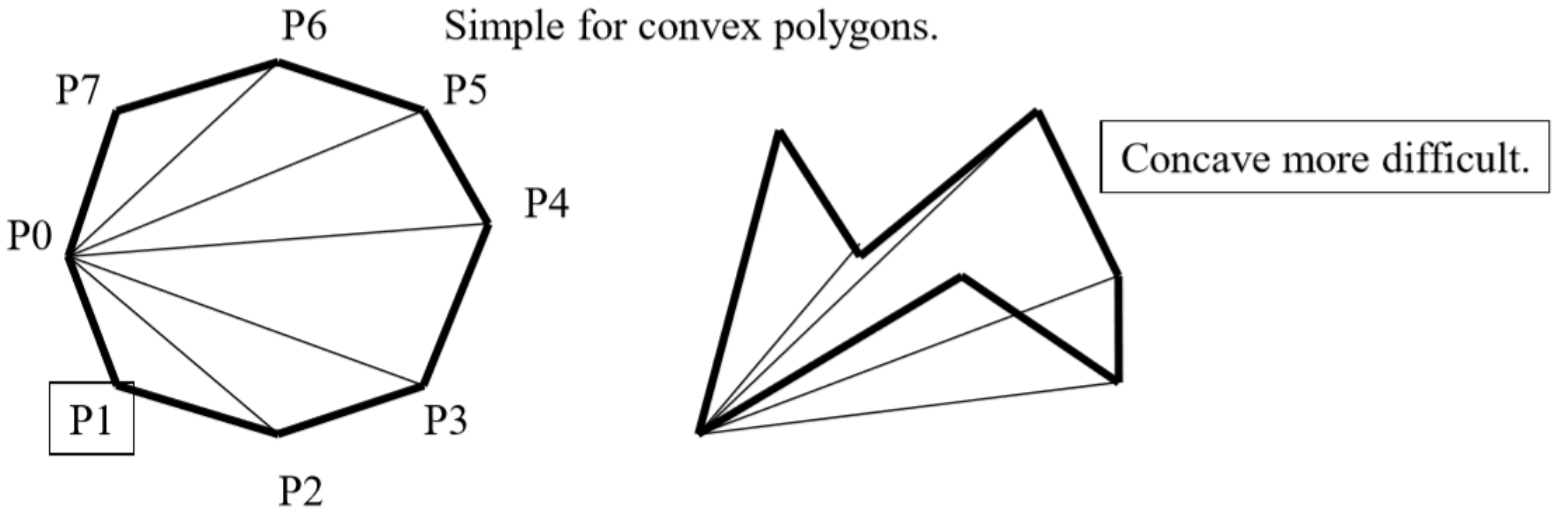
- Son convexos.
- Sus vértices son coplanares.
- No se intersecan consigo mismos.





Triángulos en vez de polígonos

La descomposición de polígonos consiste en convertirlos en un conjunto de triángulos.



Tomado de [9]

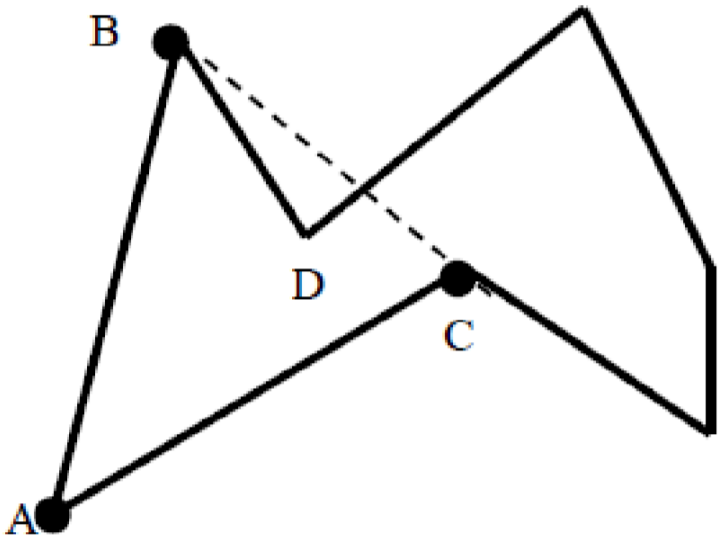
Descomposición de polígonos

Algoritmo:

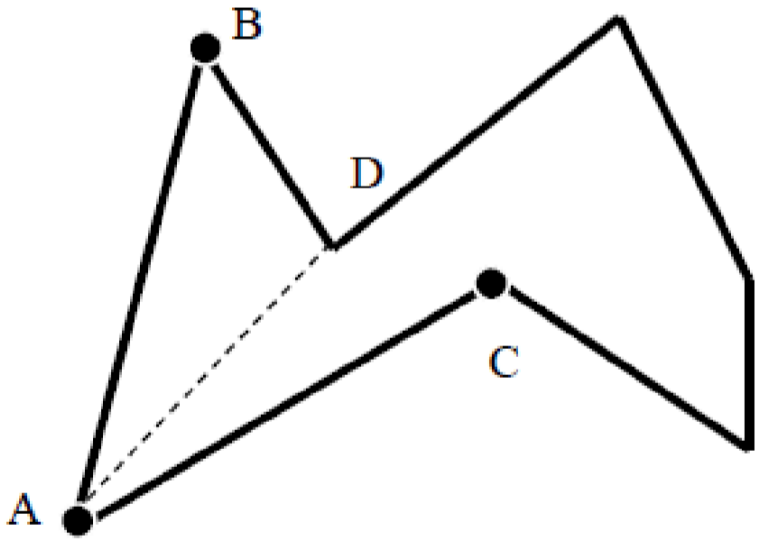
1. Partiendo del vértice ubicado más a la izquierda, formar un *posible* triángulo con sus dos vértices adyacentes.
2. Verificar que dentro del triángulo formado no existan otros vértices perteneciente al polígono.
   - Si existen, tomar el vértice interno ubicado más a la izquierda y formar el *posible* triángulo. Repetir el punto 2 hasta que no haya vértices internos.
3. Si no hay otros vértices internos, crear el nuevo triángulo.
4. Eliminar el vértice que queda desconectado de los otros vértices del polígono.
5. Repetir hasta tener un polígono con 3 vértices.

Descomposición de polígonos

Polígono interno:



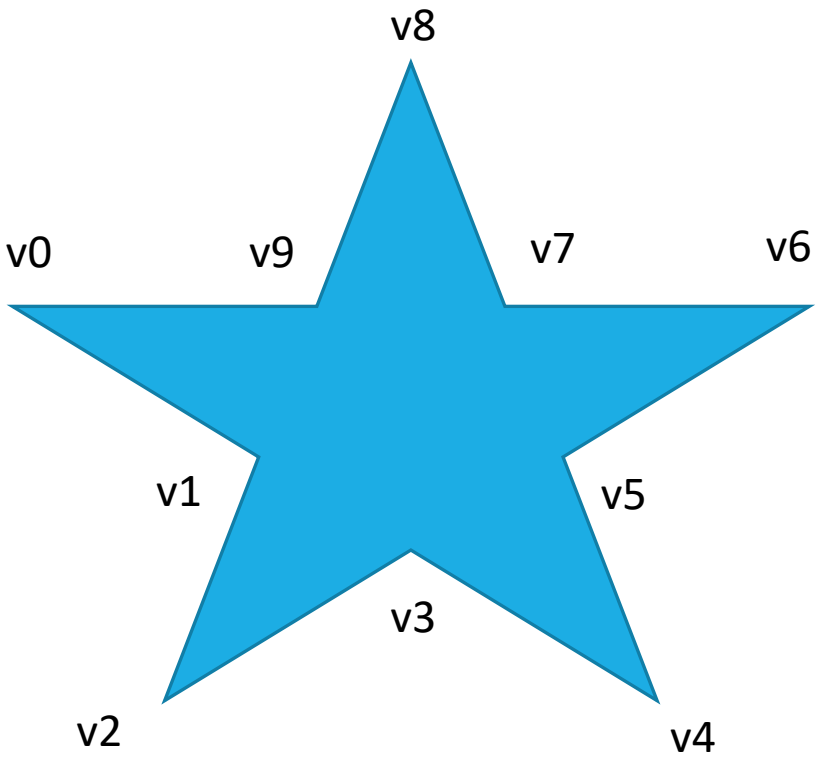
Nuevo posible triángulo:



Tomado de [9]

Descomposición de polígonos

Dado el siguiente polígono, realizar la descomposición del mismo y entregar el conjunto de triángulos final:



Ejercicio

Permiten verificar si un pixel está dentro o fuera de un triángulo.

Además permiten realizar interpolación de valores dentro del triángulo.

Trabajo en clase:

1. Investigar qué son las coordenadas baricéntricas.

2. Dadas las coordenadas de los puntos de un triángulo, hallar, imprimir y mostrar el baricentro, el incentro y el cincuncentro.

Coordenadas baricéntricas

Bibliografía

[1] Rueda, A. (2014). ***Rastreo y recorte – Primitivas 2D***. Pontificia Universidad Javeriana.

[2] Tutorials point. Polygon filling algorithm. https://www.tutorialspoint.com/computer_graphics/polygon_filling_algorithm.htm

[3] Tutorials point. Viewing & Clipping . https://www.tutorialspoint.com/computer_graphics/viewing_and_clipping.htm

[4] Gotsman, Elber, Barequet, Karni, Sheffer. Polygon Clipping – Drawing Geometry on Raster Displays. Computer Graphics. The University of Arizona. Tomado de: https://www2.cs.arizona.edu/classes/cs433/fall08/lectures/ScanC.prn.pdf

[5] Computer graphics and geometric modeling. Tomado de: http://what-when-how.com/computer-graphics-and-geometric-modeling/

[6] Breen, D. Regli, W. y Peysakohv, M. (2015). Polygon Clipping and Filling. Tomado de: https://www.cs.drexel.edu/~david/Classes/CS430/Lectures/L-05_Polygons.6.pdf

[7] Algoritmo de relleno por difusión. Tomado de: https://es.wikipedia.org/wiki/Algoritmo_de_relleno_por_difusi%C3%B3n

[8] Preenchimento de poligonos (Relleno de polígonos). Tomado de: http://gpolo.awardspace.info/fill/main.html

Bibliografía

[9] Komura, T. Lecture 6 – Rasterization. Tomado de:

Bibliografía