Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación
CC3501 - Modelación y Computación Gráfica para Ingenieros

Apunte Gráfica

Profesor: Patricio Inostroza
Autores: Julieta Coloma, Luciano Avegno y Gustavo Joyo

Índice

Raster y Color

Existen distintos modelos de color para presentar figuras, imágenes o impresiones. Por ejemplo están: [CMYK], [HSV], [YIQ], pero sin duda el más importante, y al cual todos convergen es [RGB]

Espacios de Color

- **RGB**: Espacio de color que se basa en mezclar rojo, verde y azul. También se interpreta como un sistema aditivo dado que vamos a sumar los aportes de cada componente para obtener el color final.

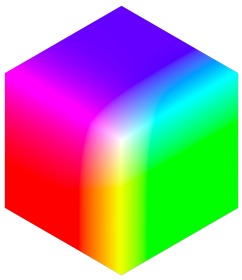


Figura 1: Cubo RGB

- **CMKY**: Este es un modelo sustractivo. Aquí el negro se encuentra al aportar todos los componentes al máximo. Se utiliza en impresoras.

Figura 2: Combinar todos los componentes da negro

- **HSV**: Presenta una versión más intuitiva de la combinación de colores (ya que rgb no es muy humano)

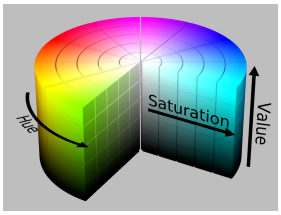


Figura 3: HSV mapeado en un cilindro

- **YIQ**: Usado en televisiones. Posee una cantidad reducida de colores.

Colores en el computador

- **Interpolación**: Dados ciertos valores, interpolación se refiere al proceso en que encontramos los valores intermedios entre nuestros puntos originales.

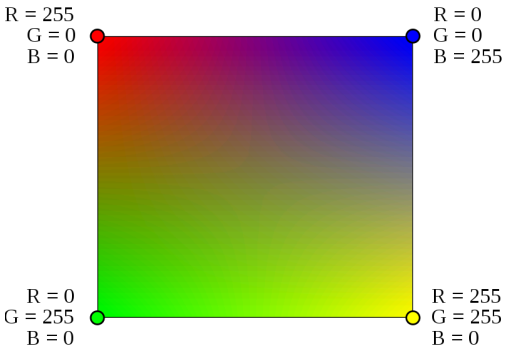


Figura 4: Interpolación de los colores, dados los 4 vértices del rectángulo

- **Almacenamiento**: ¿Cómo se almacena un color (RGB)?

Necesitamos 8 bits por componente, como son 3 componentes tenemos un total de 24 bits. Luego incluímos el canal alpha (transparencia) y nos quedamos con un total de 32 bits.

- **¿Transparencia?**

Al incluir la transparencia se obtiene la siguiente fórmula

$$c = ac_f = (1 - a)c_b$$

Donde c_f es el color de el frente, c_b el color de atras y a la transparencia o *alpha* del frente.

Raster y Vector

- **Imagen raster**: Se compone de una matriz 2D donde a cada celda se le conocerá como un pixel, son usados en formatos: png, jpg, bmp o gif.
- **Imagen vectorial**: Usan modelos paramétricos por cada figura representada donde no hay pérdida de nitidez en ninguna forma. Las letras tipográficas funcionan así, son usados en formatos: svg, eps y vrml.
- **Pantalla Raster**: Define un arreglo 2D en pixeles centrados en coordenadas enteras de forma que se pueda iluminar un pixel con setpixel(x, y).

Bresenham

Ya pero ¿Cómo se dibuja una línea con pixeles?

El Algoritmo de Bresenham es eficiente a la hora de dibujar una línea en una pantalla raster, es decir, que esté compuesta por pixeles. Este algoritmo se puede utilizar en rectas que tengan una pendiente entre 0 y 1.

Se tiene una línea recta desde (x_0, y_0) a (x_1, y_1) con pendiente $m \in [0, 1]$. Tras pintar (x_0, y_0) se quiere saber si pintar el pixel $(x_0 + 1, y_0)$ o $(x_0 + 1, y_0 + 1)$. En la figura se marcan los pixeles candidatos en verde.

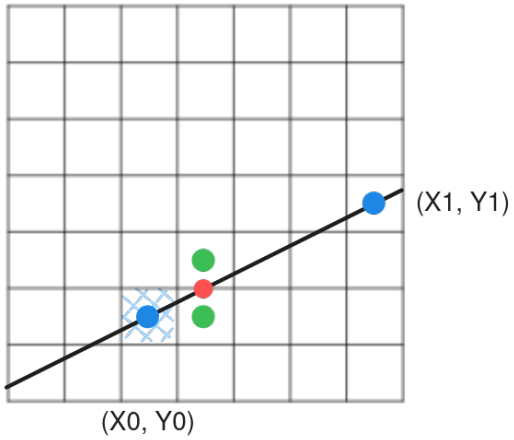


Figura 5: Pixeles candidatos en verde

Primero, tomemos la ecuación en su forma $f(x, y) = Ax + Bx + C = 0$ donde

- $A = \Delta y = y_1 - y_0$
- $B = -\Delta x = -(x_1 - x_0)$
- $C = (\Delta x)c = (x_1 - x_0)c$

Para cualquier punto de la recta $f(x, y) = 0$, $f(x, y) > 0$ para puntos sobre la recta y $f(x, y) < 0$ para puntos bajo la recta.

Evaluamos el punto medio entre y_0 e $y_0 + 1$. Si $f\left(x_0 + 1, y_0 + \frac{1}{2}\right)$ es positivo, entonces la línea se acerca más al punto superior y pintamos el pixel $(x_0 + 1, y_0 + 1)$, de los contrario pintamos $(x_0 + 1, y_0)$[1]

[1]Para una definición similar a la del profesor Patricio Inostroza lea aquí: https://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html

Para pendientes fuera del rango $[0, 1]$ utilizamos simetría

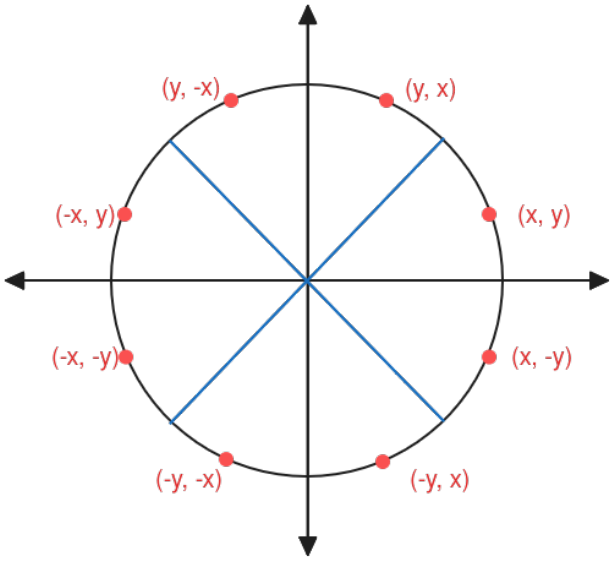


Figura 6: Uso de la simetría para aplicar el algortimo al resto de los octantes.

Bresenham en el Círculo

to-do

Transformaciones

Este documento pretende ser un punteo así que se mencionarán rápidamente las matrices de transformación.

- **Coordenadas Homogéneas**: Es un sistema de coordenadas utilizado en computación gráfica (y otras áreas) para la geometría proyectiva. Las matrices que se mencionarán a continuación están en coordenadas homogéneas.
- **Escalamiento**: Escalar por las magnitudes s_x, s_y y s_z en sus respectivos ejes.

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Rotación sobre el eje x**: Rotación de θ grados anti-reloj

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Rotación sobre el eje y**: Rotación de θ grados anti-reloj

$$\begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 1 & 0 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Rotación sobre el eje z**: Rotación de θ grados anti-reloj

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Traslación**: Trasladar el punto por las magnitudes d_x, d_y y d_z en sus respectivos ejes

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Reflexión**: La reflexión se extiende de el escalamiento pero con magnitudes negativas. Por ejemplo, la reflexión con respecto al eje Y cambia el signo del componente en X

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Shearing**

$$\begin{bmatrix} 1 & s_{xy} & s_{xz} & 0 \\ s_{yx} & 1 & s_{yz} & 0 \\ s_{zx} & s_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Composición de transformaciones**: Se pueden realizar varias transformaciónes juntas para obtener una gran matriz de transformación.

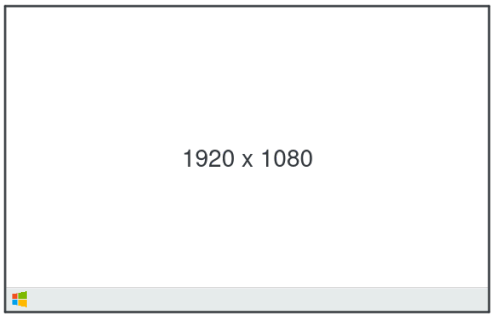
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

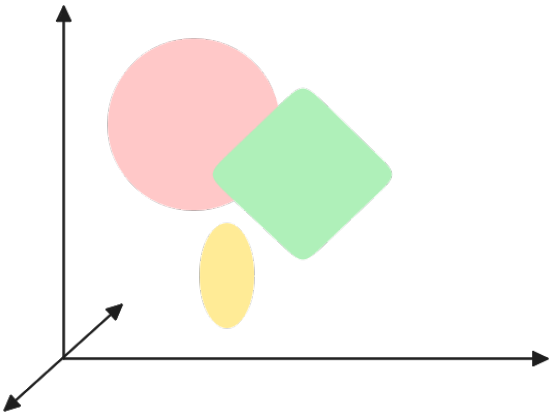
Viewing

Lo que se debe rescatar de esta sección es principalmente las diferencias entre los distintos sistemas de coordenadas que usamos en las apis (OpenGL), en las pantallas u otra forma de representar imágenes.

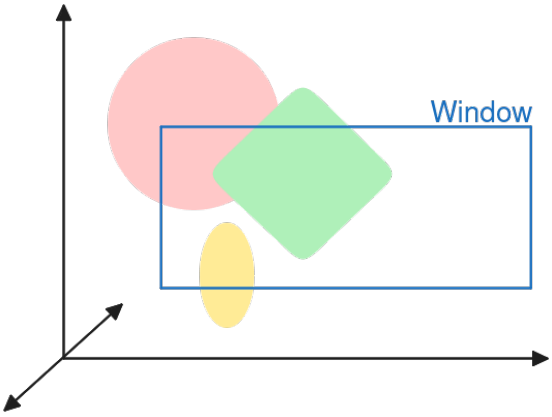
- **Coordenadas del dispositivo o Display Coordinates (DC)**: En este sistema se especificara en que parte de la pantalla mostrar información.



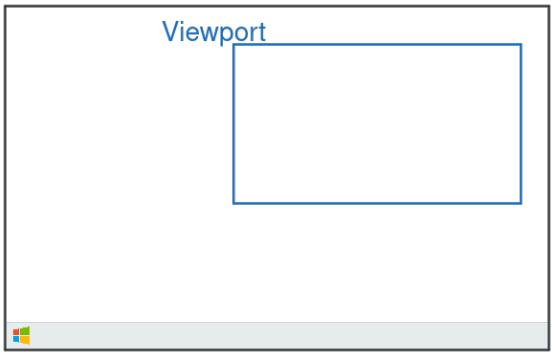
- **Coordenadas del mundo o World Coordinates (WC)**: En este sistema se especifican los objetos que queremos mostrar y en ellas se basan las transformaciones.



- **Window**: Sección de WC que quermos mostrar



- **Viewport**: Seccion de DC en donde queremos mostrar.



- **Coordenadas normalizadas o Normalized device coordinates (NDC)**: Es un sistema de coordenadas independiente de la pantalla del dispositivo. Existe en el rango de $[-1, 1]$ y es un punto medio entre WC y DC. Su objetivo es ser un intermediario para mostrar la información en cualquier pantalla.

Window a Viewport

Transformar un punto del window (x_w, y_w) a un punto en el viewport (x_v, y_v)

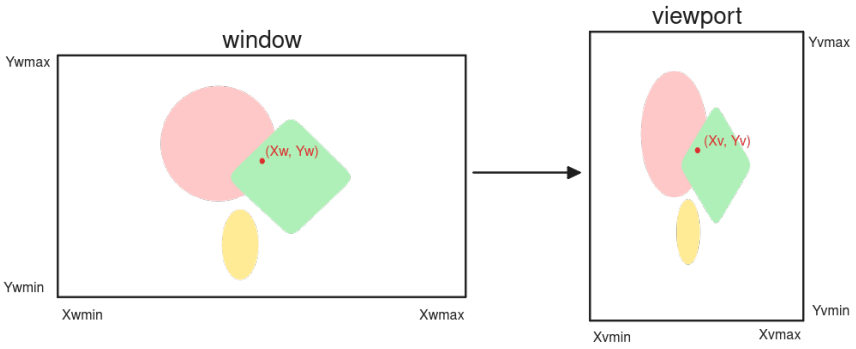


Figura 11: Transformación de window a viewport.

Se tiene que

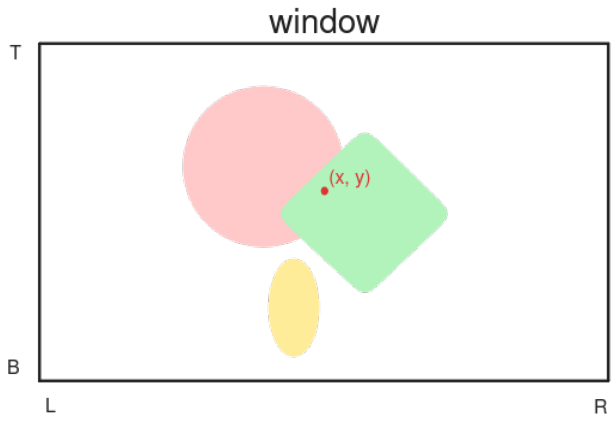
$$X_v = (X_w - X_{w\,\min}) \cdot \frac{X_{v\,\max} - X_{v\,\min}}{X_{w\,\max} - X_{w\,\min}}$$

$$Y_v = (Y_w - Y_{w\,\min}) \cdot \frac{Y_{v\,\max} - Y_{v\,\min}}{Y_{w\,\max} - Y_{w\,\min}}$$

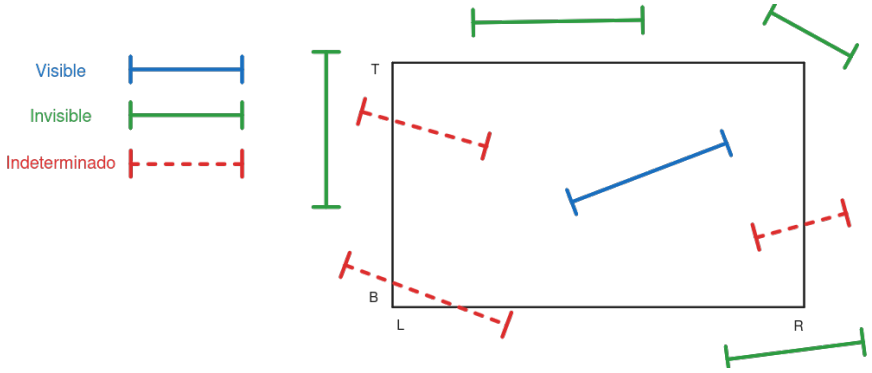
Clipping

Se quiere identificar los elementos que serán visibles en el window y los que no.

- **Clipping sobre un punto**: Es simple. Dado los extremos del window L, R, T y B, (x, y) es visible si $L \leq x \leq R$ y $B \leq y \leq T$



- **Clipping sobre un segmento**: Primero podemos determinar los segmentos que si o si son invisibles, esto es si cumple alguno de los siguientes:
  - x_1 y $x_2 < L$
  - x_1 y $x_2 > R$
  - y_1 e $y_2 < B$
  - y_1 e $y_2 > T$



¿Qué se hace con los segmentos indeterminados?

- **Algoritmo de Cohen-Sutherland para clipping**: Se divide la región en 9 secciones y a cada seccion se le asigna un código de 4 bits.

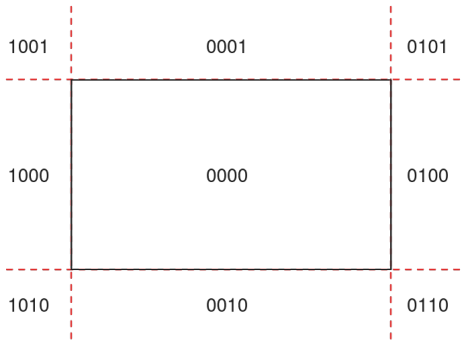


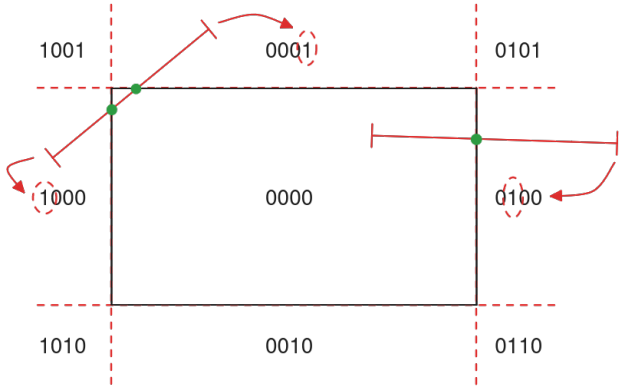
Figura 14: Secciones segun Cohen-Sutherland

Cada bit hace referencia a si se encuentra a la izquierda, derecha, abajo y arriba del window. Por ejemplo, las secciones de la izquierda tiene el primer bit encendido. Luego, tenemos que:

- Si ambos extremos tienen código `0000` entonces es un segmento visible.
- Si el resultado de la operación bitwise `&` entre ambos códigos es distinta a 0, es un segmento no visible.

```
if extremo1 == 0 and extremo2 == 0:
  segmento es visible
else if extremo1 & extremo2 != 0:
  segmento no-visible
else:
  indeterminado
```

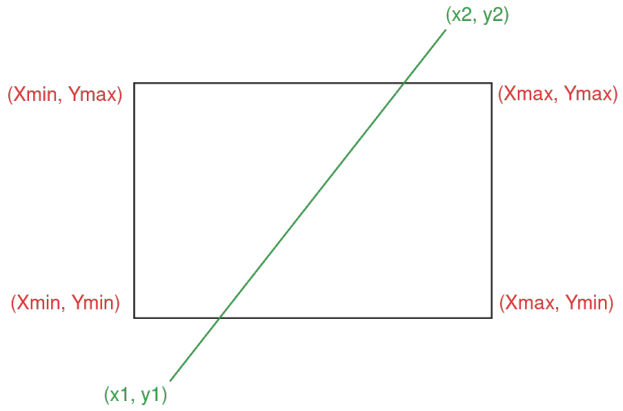
Ahora se trabaja con los segmentos indeterminados. Si se analizan los códigos de sus extremos se puede identificar con qué borde intersecta el segmento.



En la figura se ve que el segmento de la derecha intersecta con el borde de la derecha, tal y como indica su código. Se deben obtener las intersecciones con los bordes y de esa forma se define el segmento interior como aquel que es visible. Basta con plantear las ecuaciones de recta para el segmento, reemplazar x o y con el borde intersectado y obtener el punto completo.

- **Algoritmo de Laing-Barsky para clipping**: Este argumento se basa en tomar la parametrización de la recta y acotarla a su segmento visible. Se toman las ecuaciones paramétricas del segmento que va de (x_1, y_1) a (x_2, y_2).

$$x = x_1 + t(x_2 - x_1) = x_1 + t\Delta x$$
$$y = y_1 + t(y_2 - y_1) = y_1 + t\Delta y$$



Ya se tenía que para un punto, se podía concluir que era visible si cumplia con:

$$x_{\min} \leq x \leq x_{\max},\ y_{\min} \leq y \leq y_{\max}$$
$$x_{\min} \leq x_1 + t\Delta x \leq x_{\max},\ y_{\min} \leq y_1 + t\Delta y \leq y_{\max}$$

Se separan las dos desigualdades en las 4 siguientes

- $t\Delta x \geq x_{\min} - x_1$,
- $t\Delta x \leq x_{\max} - x_1$,
- $t\Delta y \geq y_{\min} - y_1$,
- $t\Delta y \leq y_{\max} - y_1$

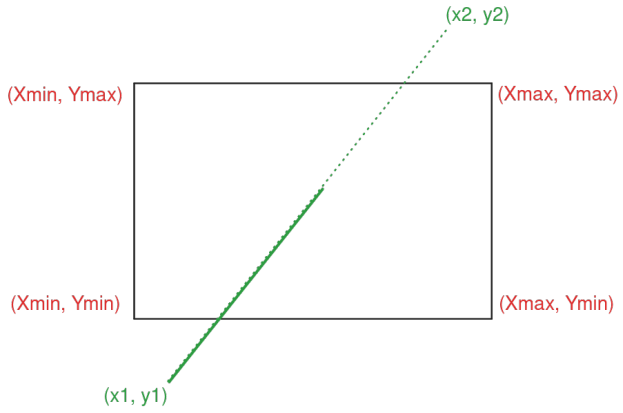
Expresadas todas de la forma $\leq$:

- $-t\Delta x \leq x_1 - x_{\min}$,
- $t\Delta x \leq x_{\max} - x_1$,
- $-t\Delta y \leq y_1 - y_{\min}$,
- $t\Delta y \leq y_{\max} - y_1$

Luego, se expresan de la forma general $t \cdot p_k \leq q_k$, donde $k = 1, 2, 3$ o 4 para cada borde izquierdo, derecho bajo y alto en ese orden.

- $p_1 = -\Delta x,\ q_1 = x_1 - x_{\min},\ t = \frac{q_1}{p_1}$
- $p_2 = \Delta x,\ q_2 = x_{\max} - x_1,\ t = \frac{q_2}{p_2}$
- $p_3 = -\Delta y,\ q_3 = y_1 - y_{\min},\ t = \frac{q_3}{p_3}$
- $p_4 = \Delta y,\ q_3 = y_{\max} - y_1,\ t = \frac{q_4}{p_4}$

Ahora, en una parametrización, si se toma $t \in [0, 0.5]$ de dibujaría hasta la mitad de la recta



De esta misma forma, lo que se quiere es modificar los valores t_1 y t_2 en $t \in [t_1, t_2]$ para dibujar exactamente el segmento visible de la recta. Dadas las ecuaciones anteriores, el algoritmo prosigue así:

1. Para cada k evaluamos las 4 inecuaciones.
2. Si $p_k = 0$ la recta es paralela al borde k.
   - Luego si $q_k < 0$ la recta está completamente fuera del window y se termina aquí el algoritmo.
   - Pero si $q_k \geq 0$ la linea está dentro del window pero paralela al borde k, no se ajusta ni t_1 ni t_2 para este k.
3. Si $p_k < 0$ se debe actualizar el primer valor t_1 a

$$t_1 = \max\left(0, \frac{q_k}{p_k}\right)$$

4. Si $p_k > 0$ se debe actualizar el segundo valor t_2 a

$$t_2 = \min\left(1, \frac{q_k}{p_k}\right)$$

5. Luego de actualizar se debe evaluar lo siguiente:
   - Si $t_1 > t_2$ la recta se rechaza.
   - Si $t_1 > 0$ cambian los valores del inicio de la recta, es decir cambian x_1 e y_1

$$x_1^{\text{new}} = x_1 + t_1 \Delta x$$
$$y_1^{\text{new}} = y_1 + t_1 \Delta y$$

   - Si $t_2 < 1$cambian los valores del fin de la recta, es decir cambian x_2 e y_2

$$x_2^{\text{new}} = x_2 + t_2 \Delta x$$
$$y_2^{\text{new}} = y_2 + t_2 \Delta y$$

Así se termina con la parametrización de solo el segmento visible de la recta.

- **Algoritmo de Sutherland-Hodgaman para clipping**

to-do

- **Algoritmo de Weiler-Atherton para clipping**

to-do

Gráfo de escena

to-do

Proyecciones

to-do

Splines

to-do

Iluminación

to-do

Visibilidad

to-do

Fractales?

to-do