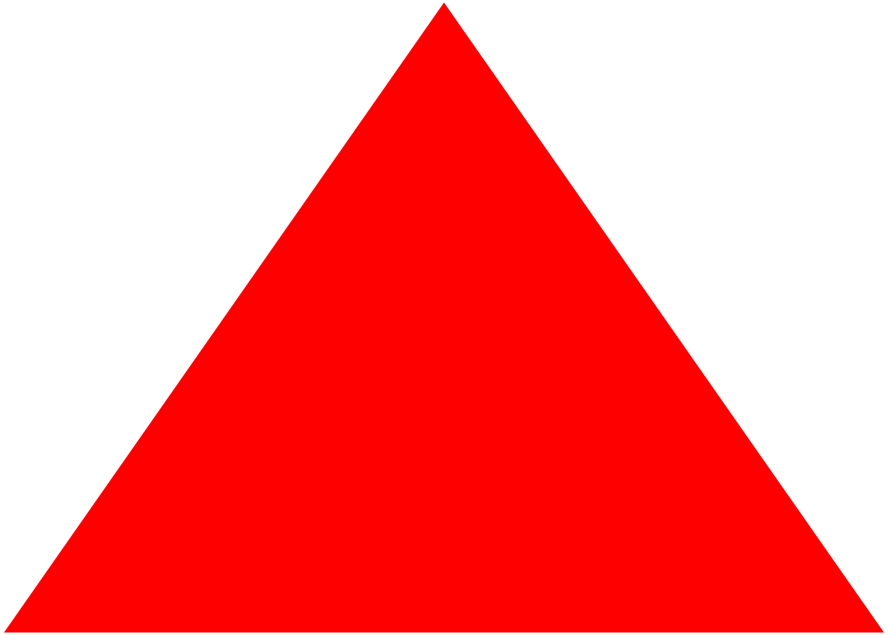
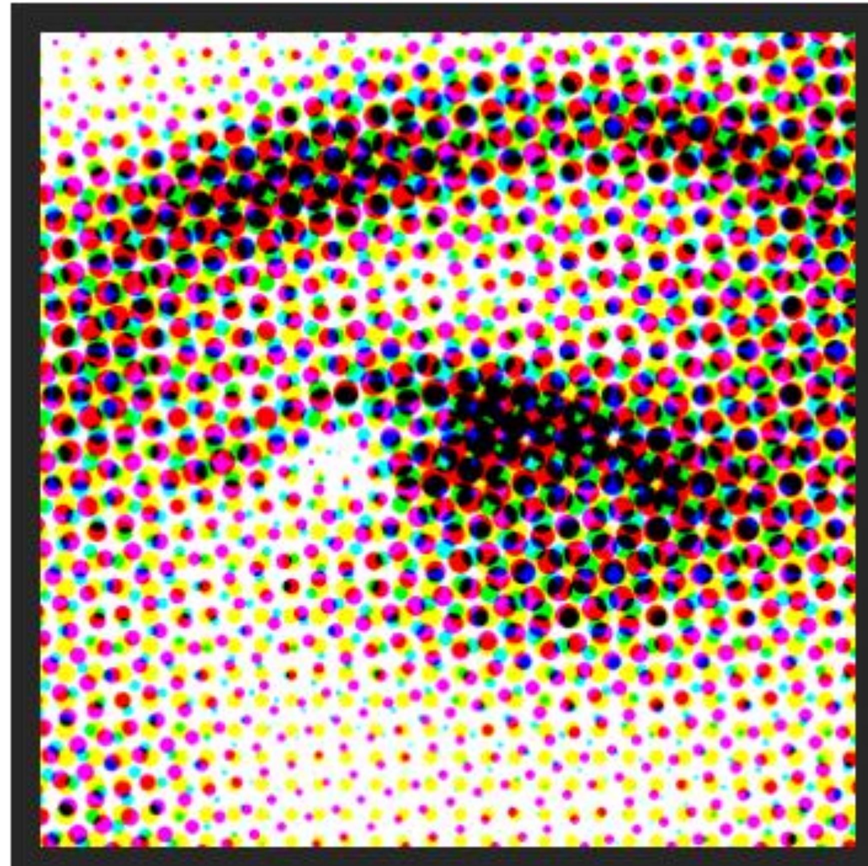
Computação Gráfica

Aula 2: Desenhando Triângulos

Desenhando Triângulos



Como as imagens são apresentadas

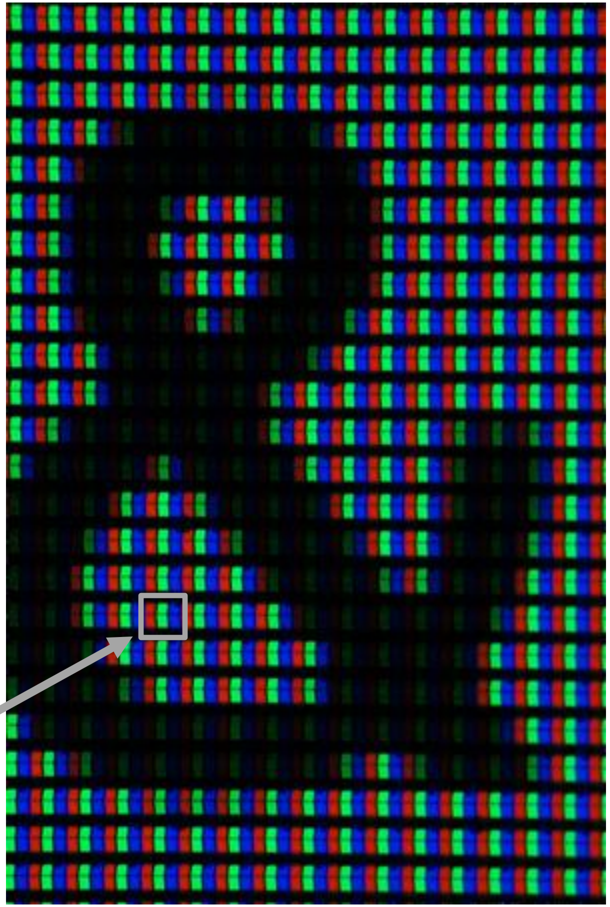


Impressão colorida : CMYK com meio-tom(half-tone)

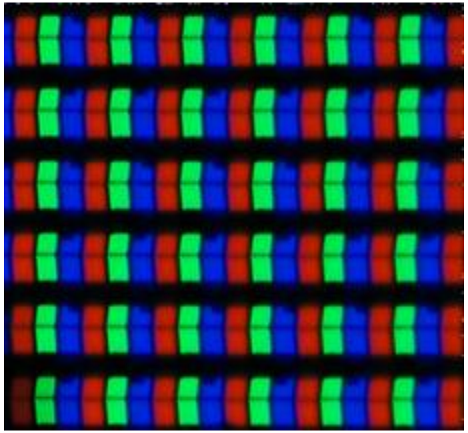
Insper

Telas / Diplays

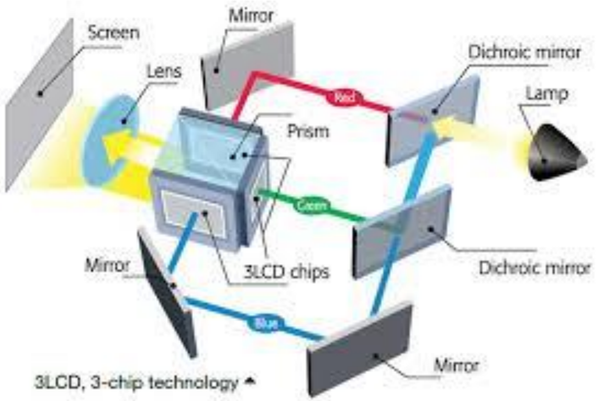
As telas são compostas de pixels
Iremos simplificar como quadradinhos
Cada "quadradinho" vai ter sua cor



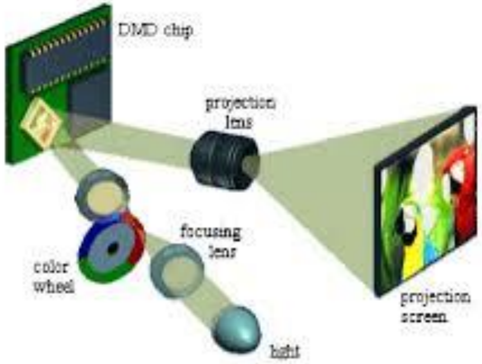
Tecnologias para gerar pixels



telas convencionais



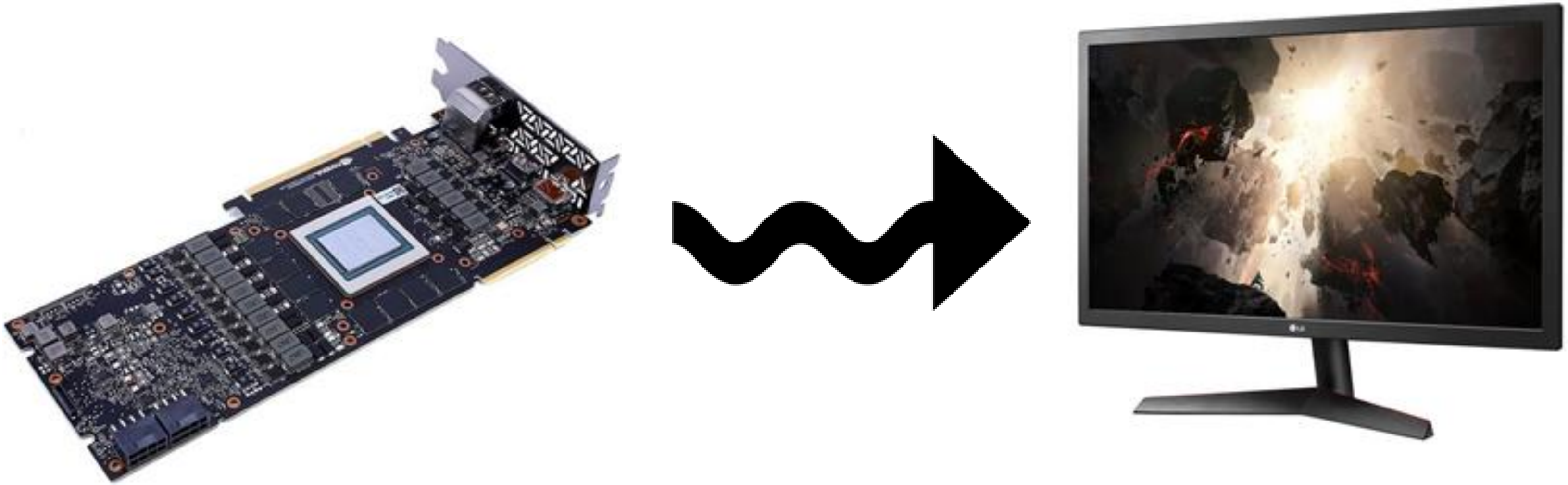
projeção 3 chip



projeção single chip

Framebuffer

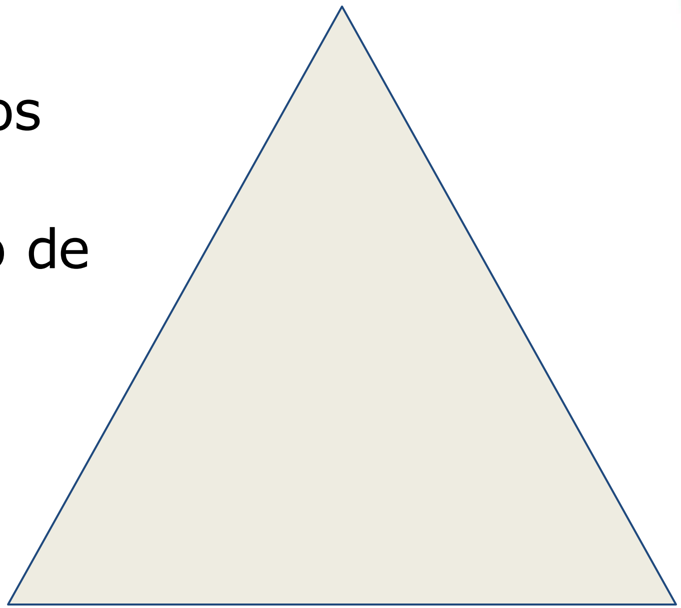
Memória na placa gráfica que será usada para produzir a imagem no monitor que estiver conectado.



Triângulo – Primitiva Fundamental

Por que triângulos?

- Polígono mais simples possível
- Outros polígonos podem ser subdivididos em triângulos
- Podemos nos focar em otimizar um tipo de operação



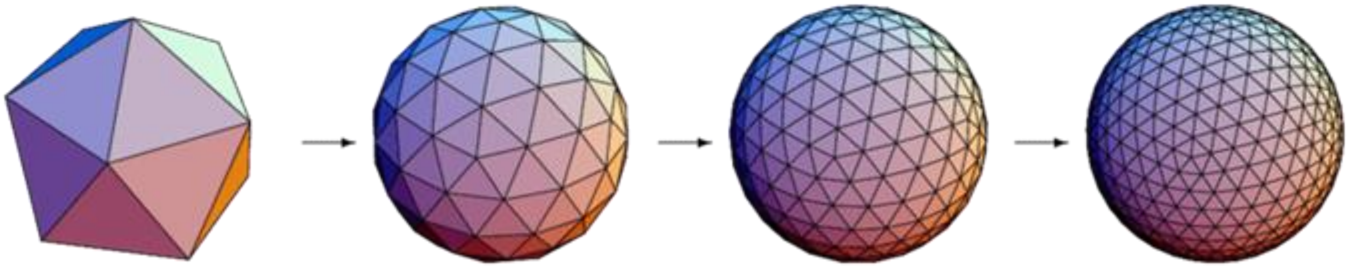
Triângulos têm propriedades únicas:

- São sempre planos
- Tem um interior bem definido
- Interpolar valores no seu interior é simples

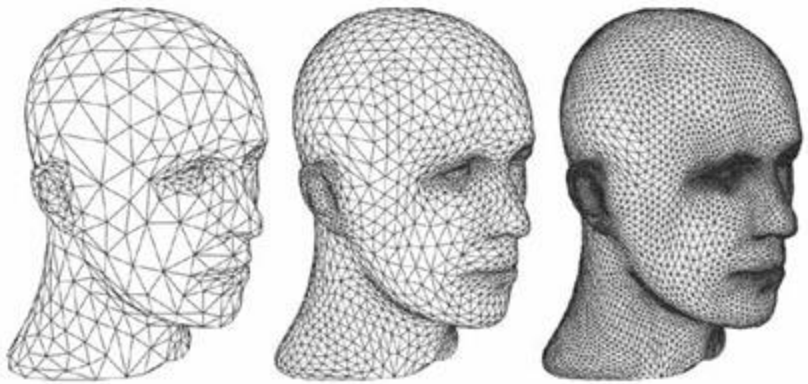
Insper

Triângulos (High e Low Poly Count)

Os triângulos são um dos blocos básicos para criar formas e superfícies geométricas mais complexas.



por Martin Styner



Low Poly e High Poly

Star Fox / SNES (1993)

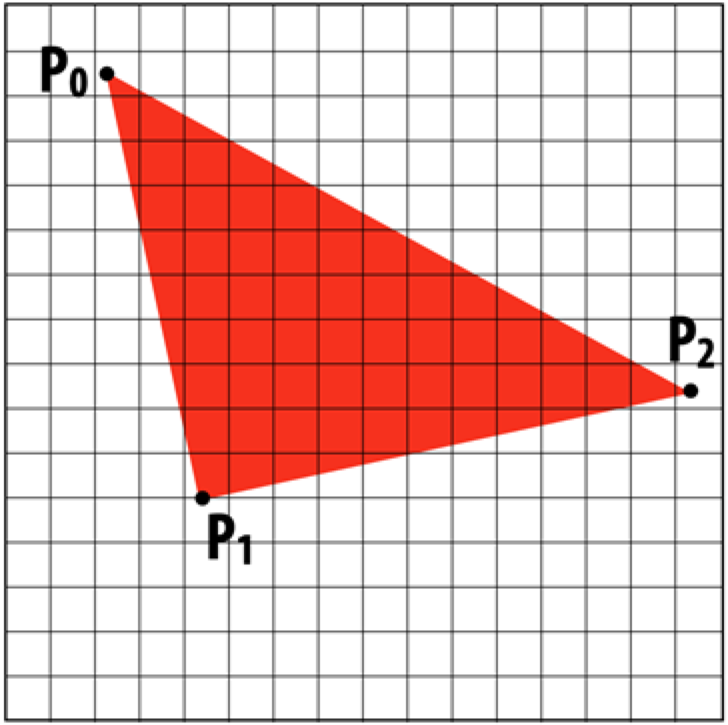
Unreal 5 / PS5 (2019)

Desenhando um Triângulo no Framebuffer

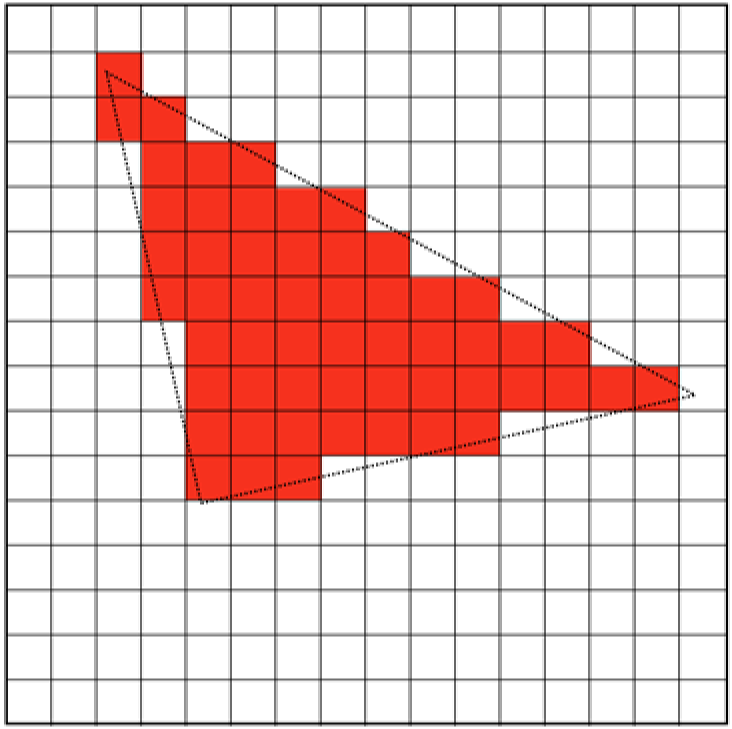
Processo que chamamos de: Rasterização (triangle rasterization)

Entrada:
Posições projetadas dos vértices do triângulo

pixels

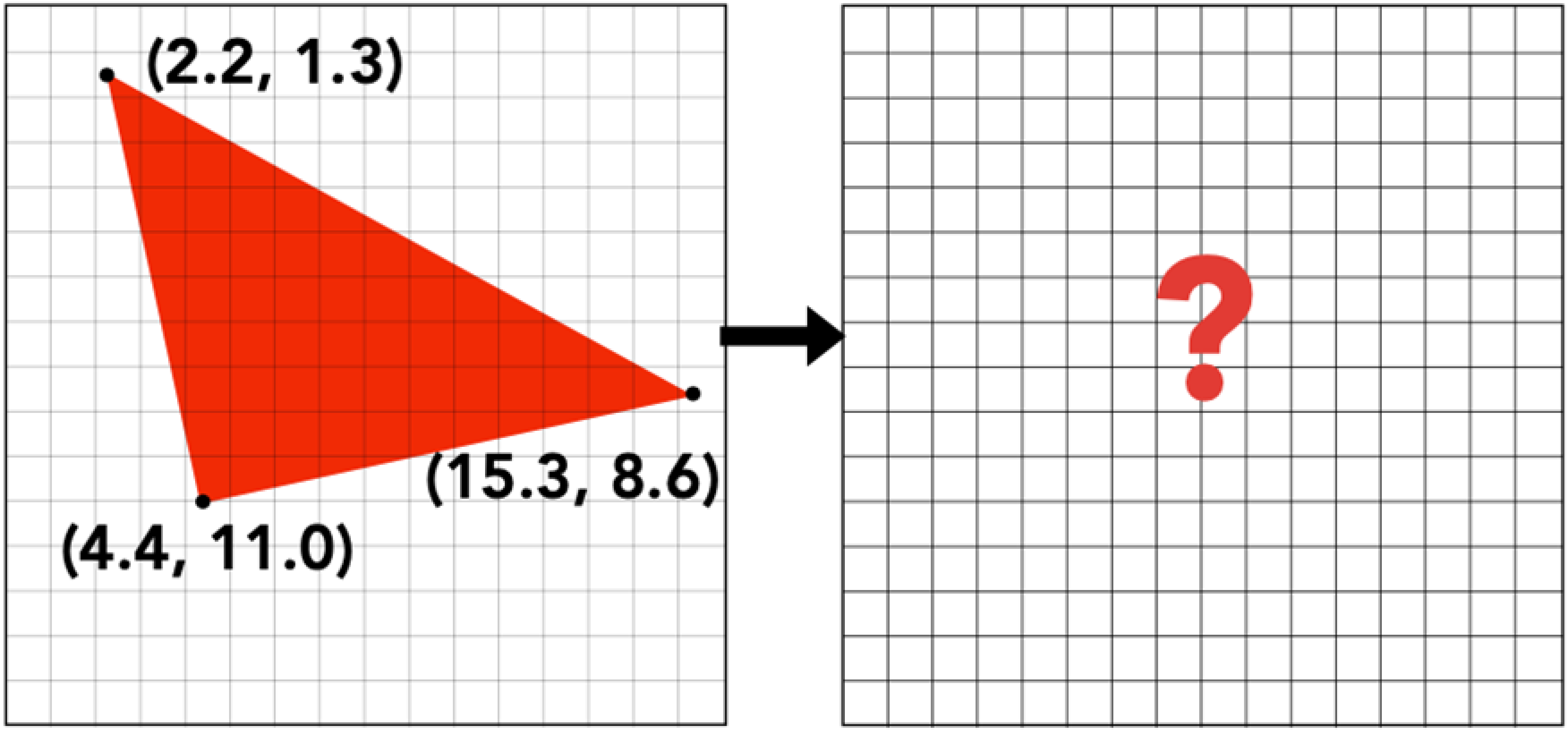


Saída:
conjunto de pixels coloridos pelo triângulo



Insper

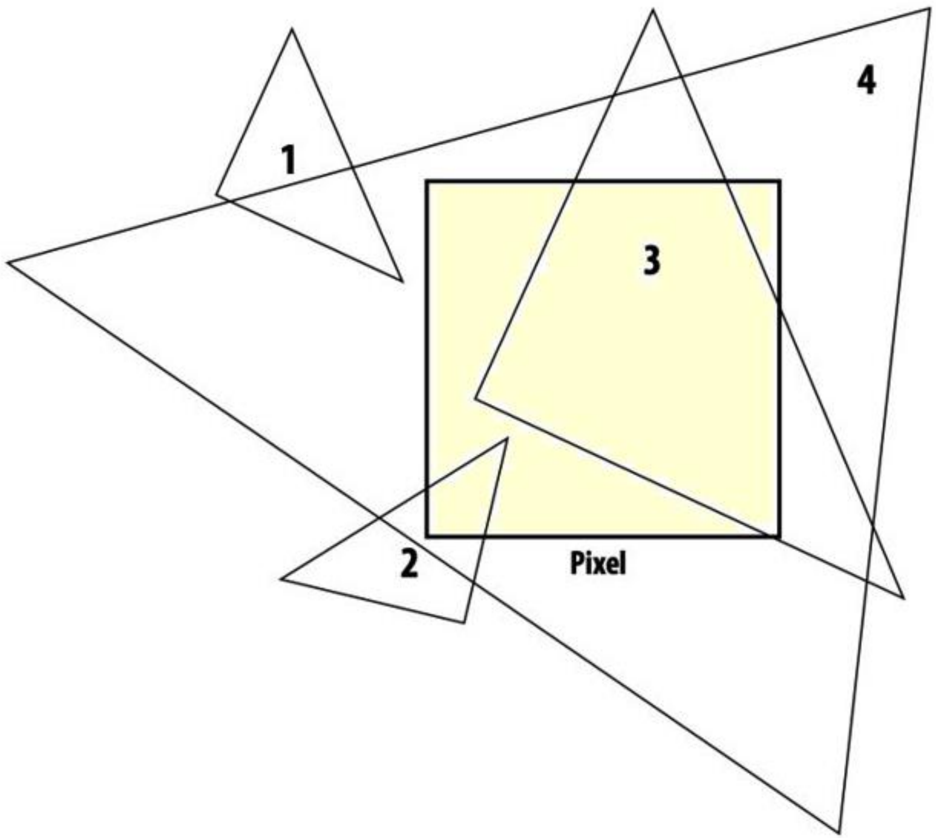
Quais valores os pixels terão após renderização



Alguns pixels são cobertos pelo triângulo outros não.

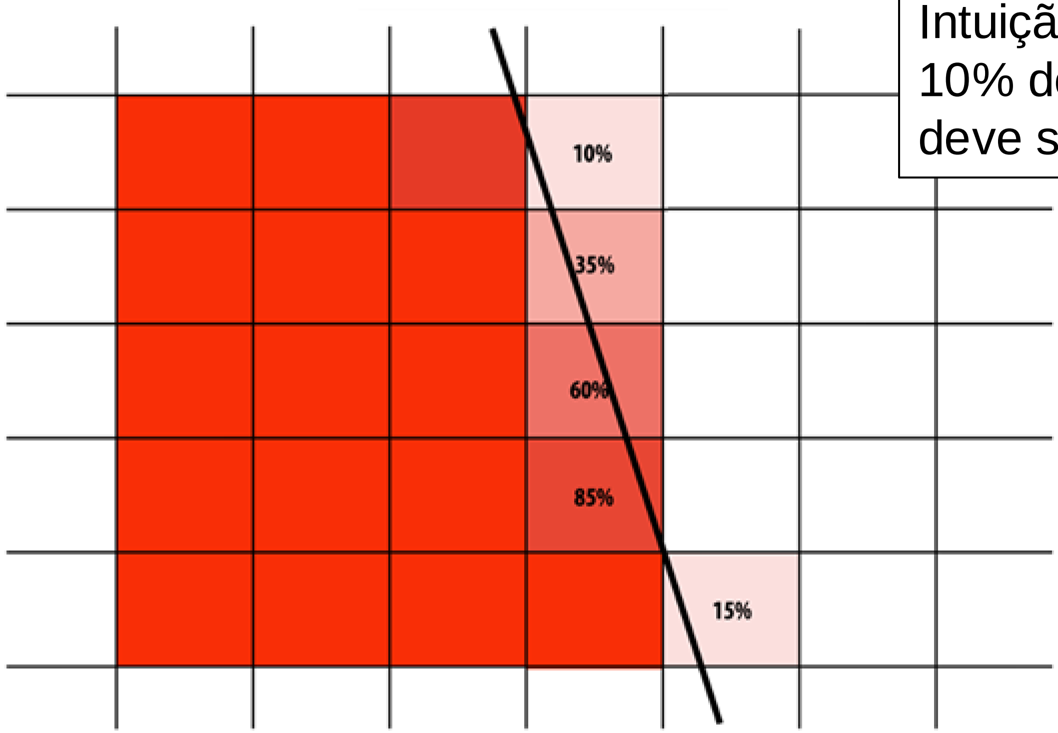
O que significa cobrir um pixel por triângulos?

Pergunta: quais triângulos "cobrem" este pixel?



Pintando os pixels sobre as arestas

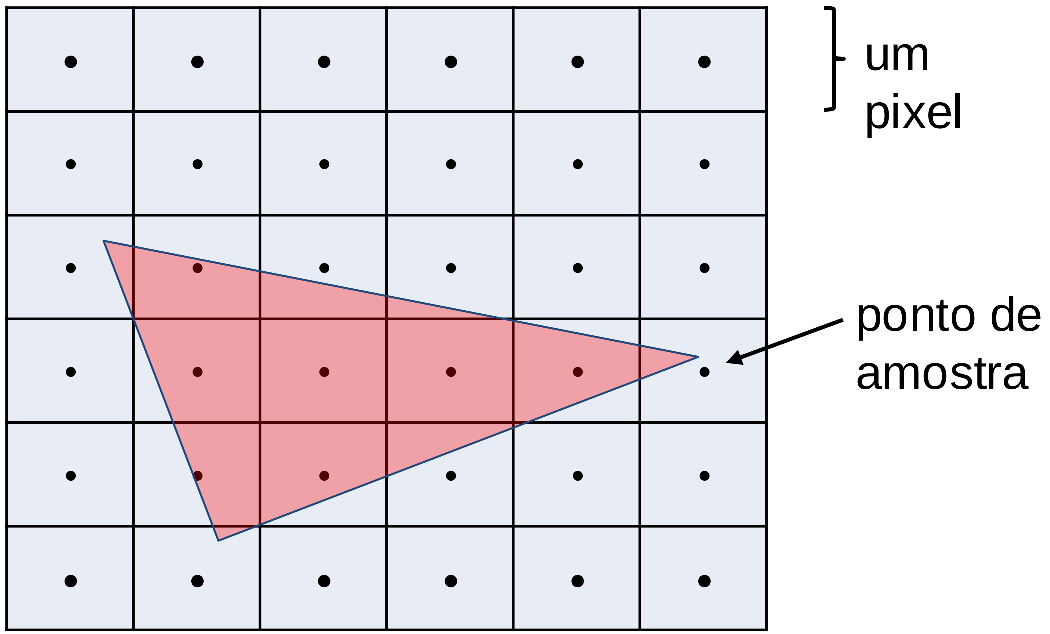
Uma opção: calcular a fração da área de pixel coberta pelo triângulo e, em seguida, colorir o pixel de acordo com essa fração.



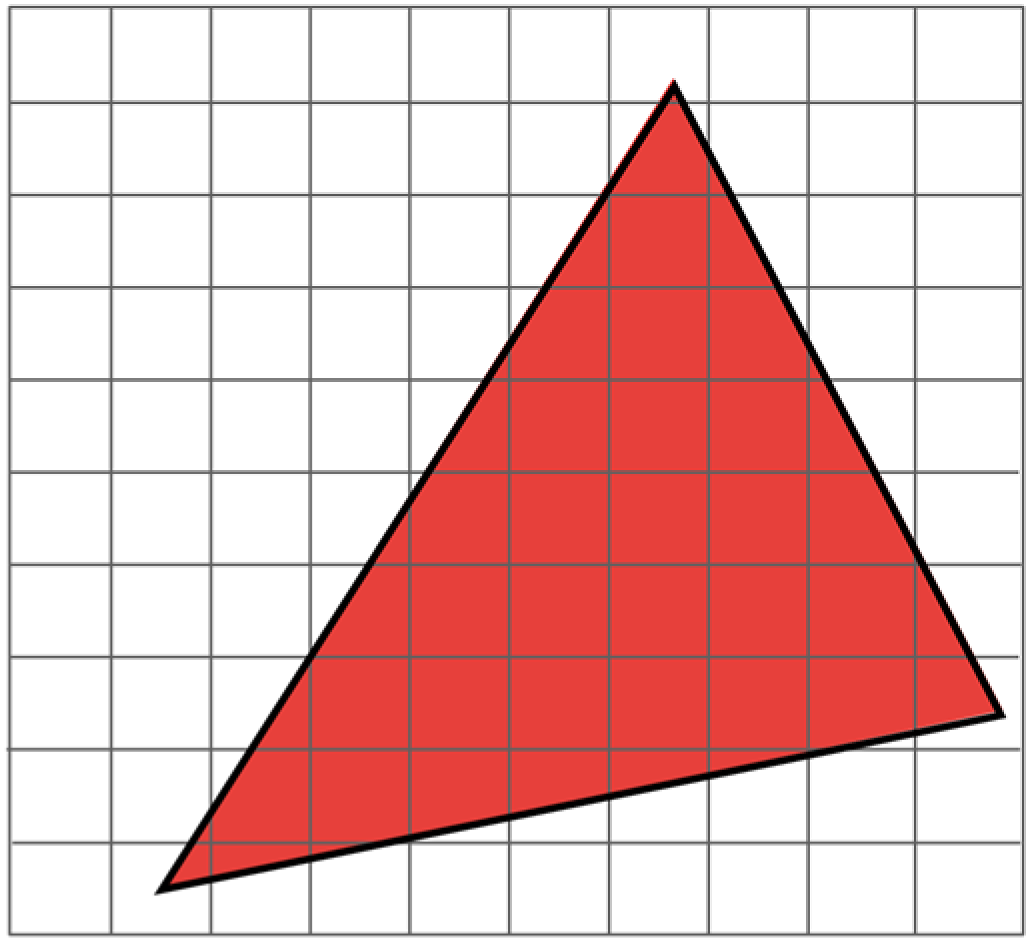
Intuição: se o triângulo cobre 10% do pixel, então o pixel deve ser 10% vermelho?

Desenhando os pixels dos triângulos

Uma forma de saber a cor do pixel é amostrando um ponto do pixel para coletar a cor do objeto.



Desenhando um triângulo por amostragem 2D



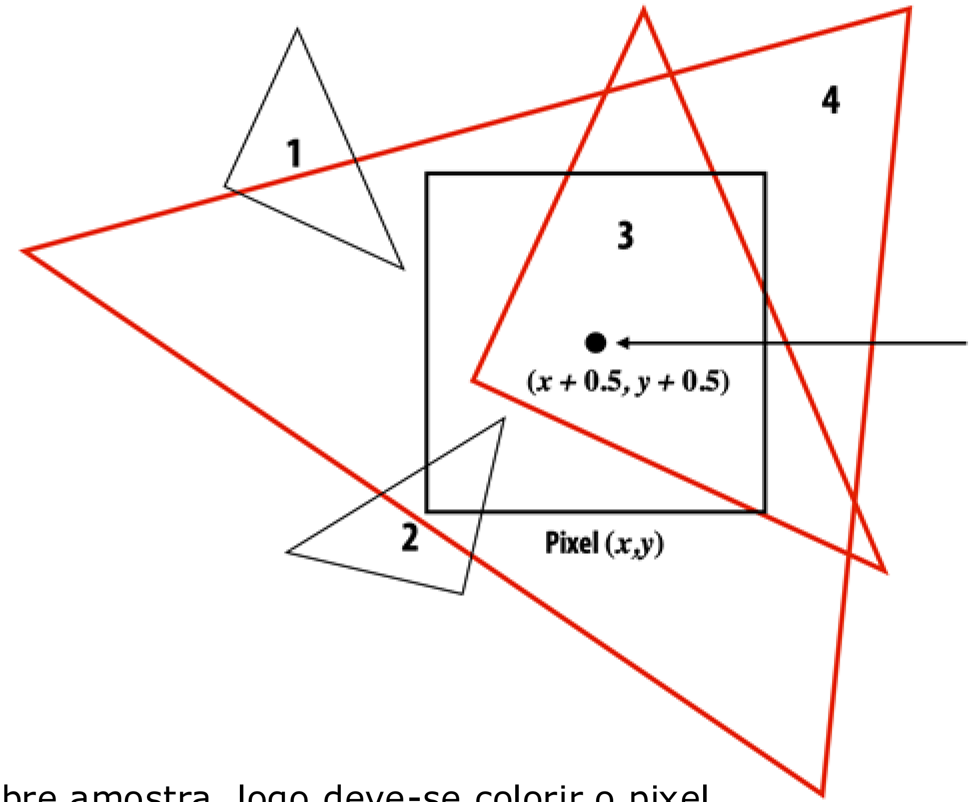
Definindo uma função binária

$$\textbf{inside(tri, x, y)} = \begin{cases} \textbf{1 se (x,y) dentro do triângulo} \\ \textbf{0 caso contrário} \end{cases}$$

tri = triângulo

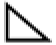
Nas APIs gráficas esse teste pode ser chamado de inside-outside test ou ainda coverage test.

Amostrando a função binária

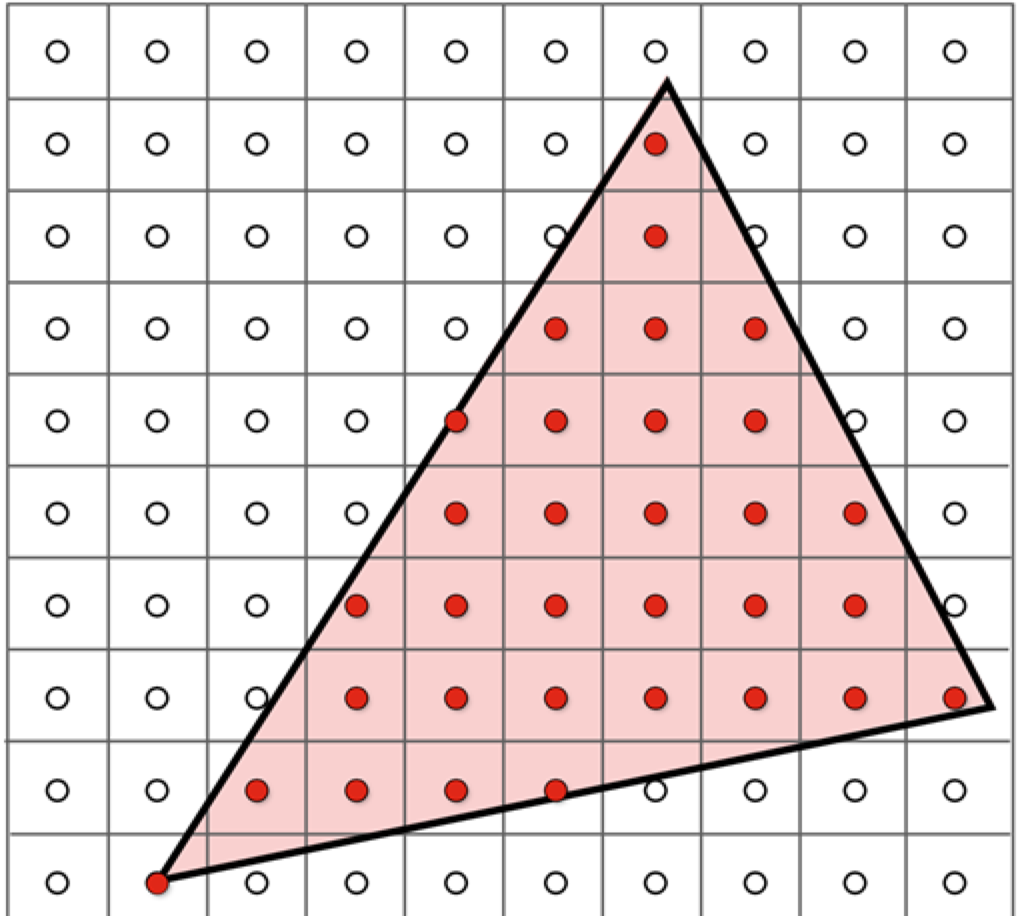


Escolhi o centro do pixel para ser o local da amostragem

 = triângulo cobre amostra, logo deve-se colorir o pixel

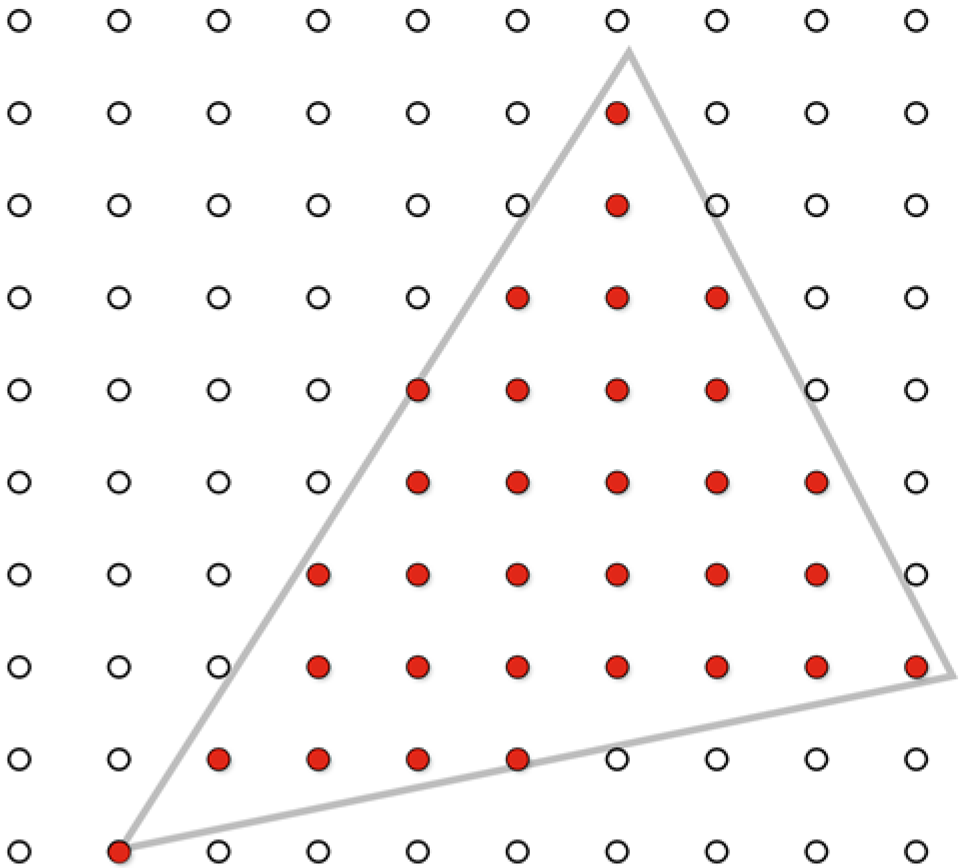
 = triângulo não cobre a amostra, não colorir o pixel

Amostras cobrindo o centro do pixel



Insper

Amostras identificando o triângulo



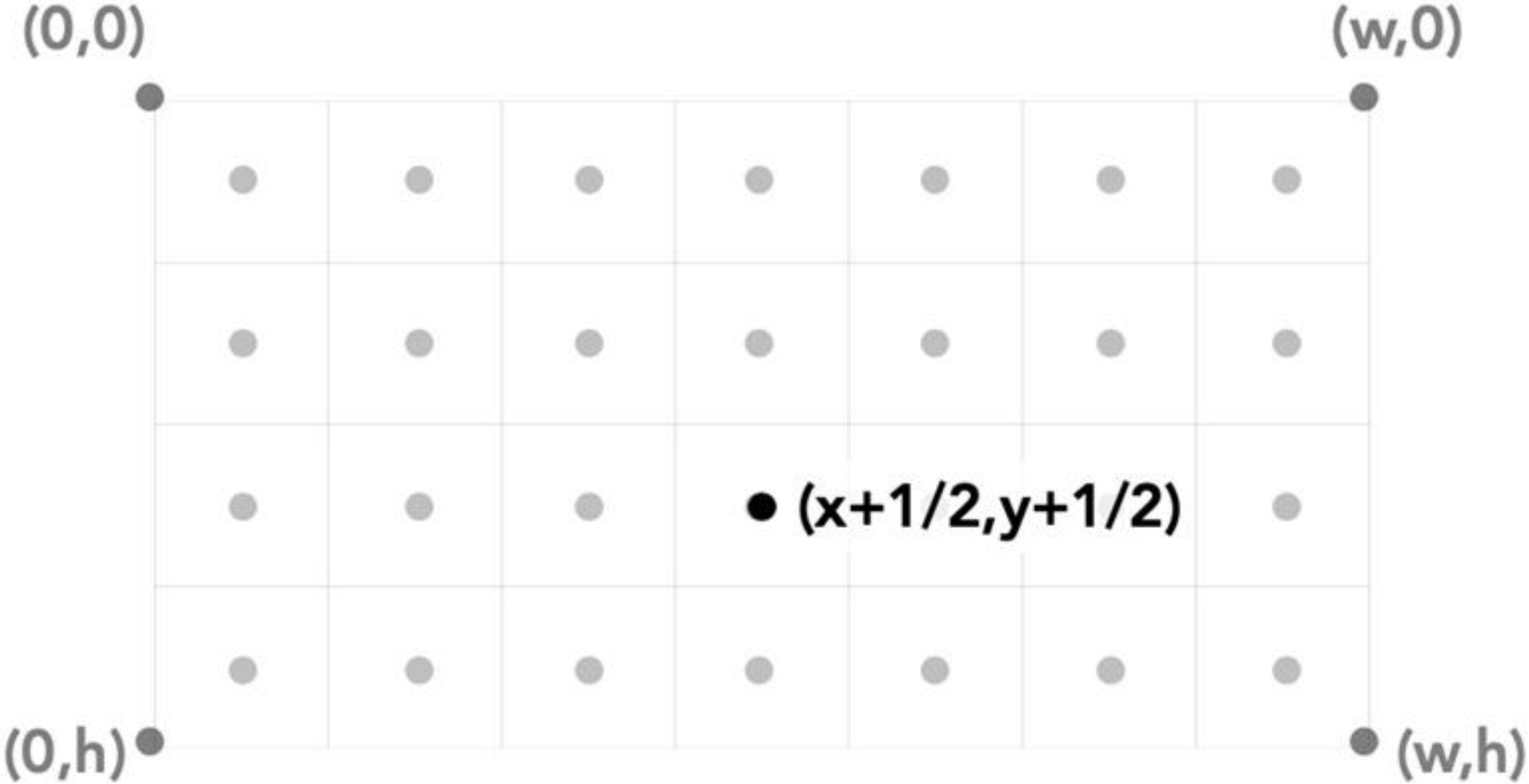
Insper

Rasterização

Amostrando os pixels de uma imagem

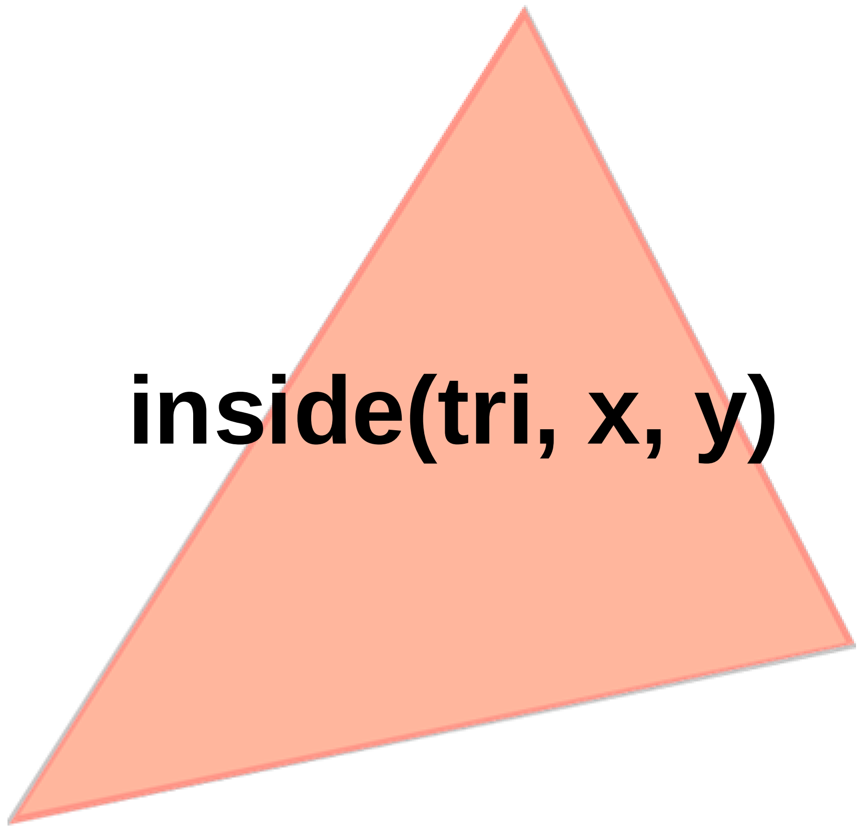
```
for (int x = 0; x < xmax; x++)
    for (int y = 0; y < ymax; y++)
        image[x][y] = inside(tri, x + 0.5, y + 0.5);
```

Detalhes: Localização da Amostra



Localização da amostra para pixel (x, y)

Avaliando se dentro do triângulo

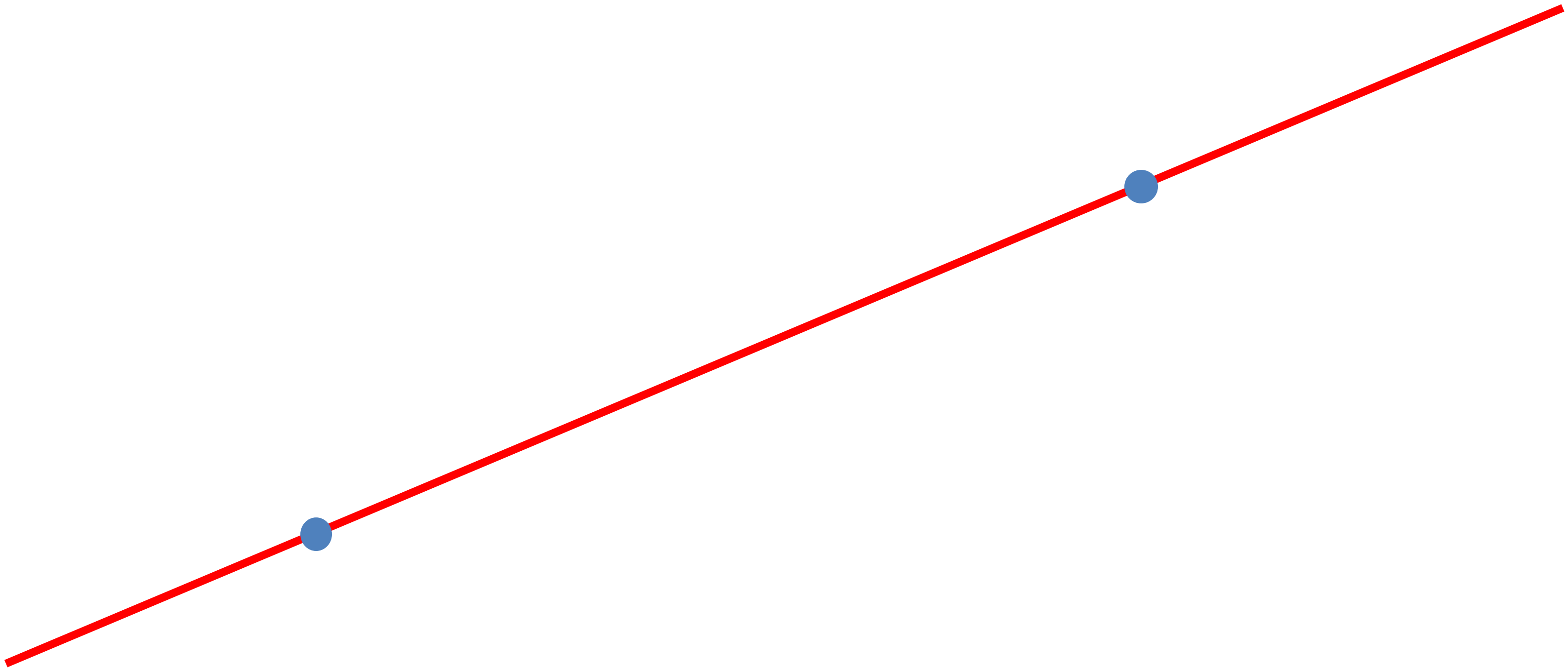


Mas como saber se dentro do triângulo?

Antes disso. Como definimos um triângulo?

Reta

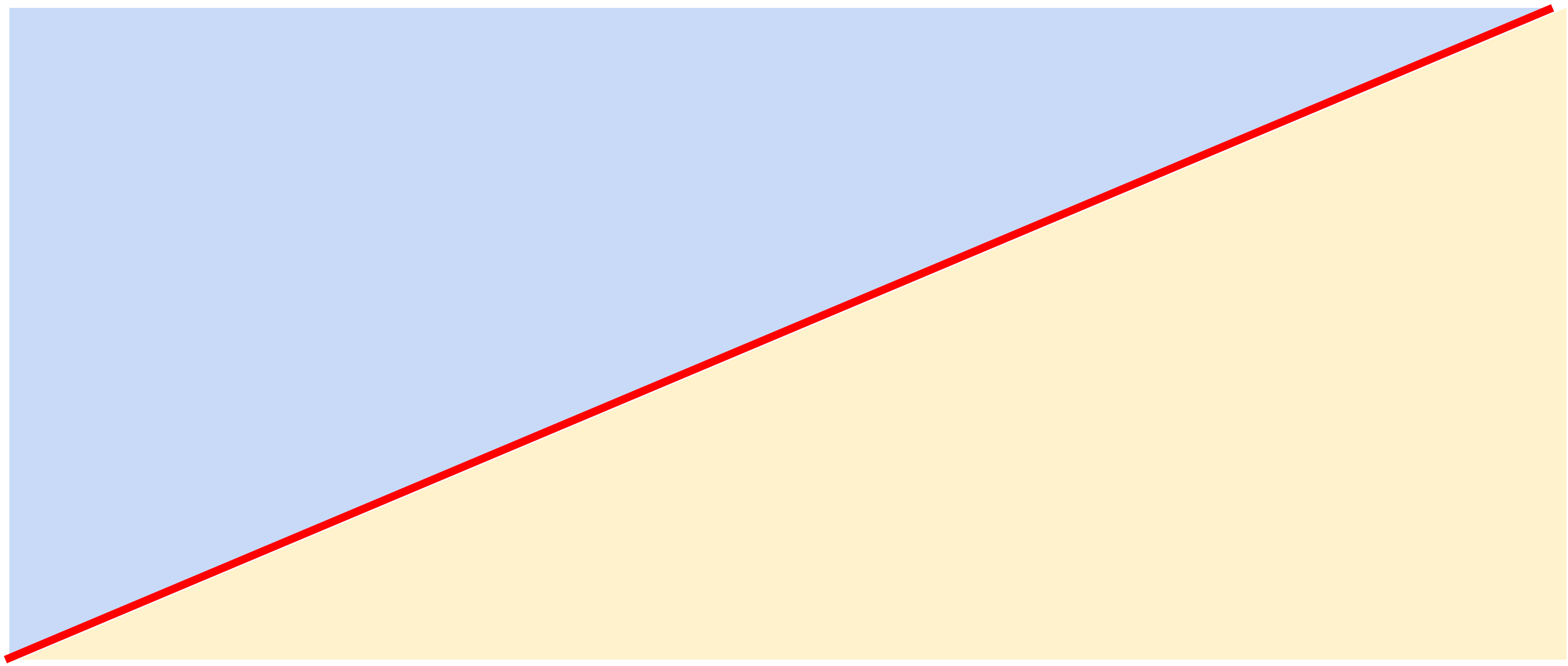
Dois pontos definem uma reta.



Insper

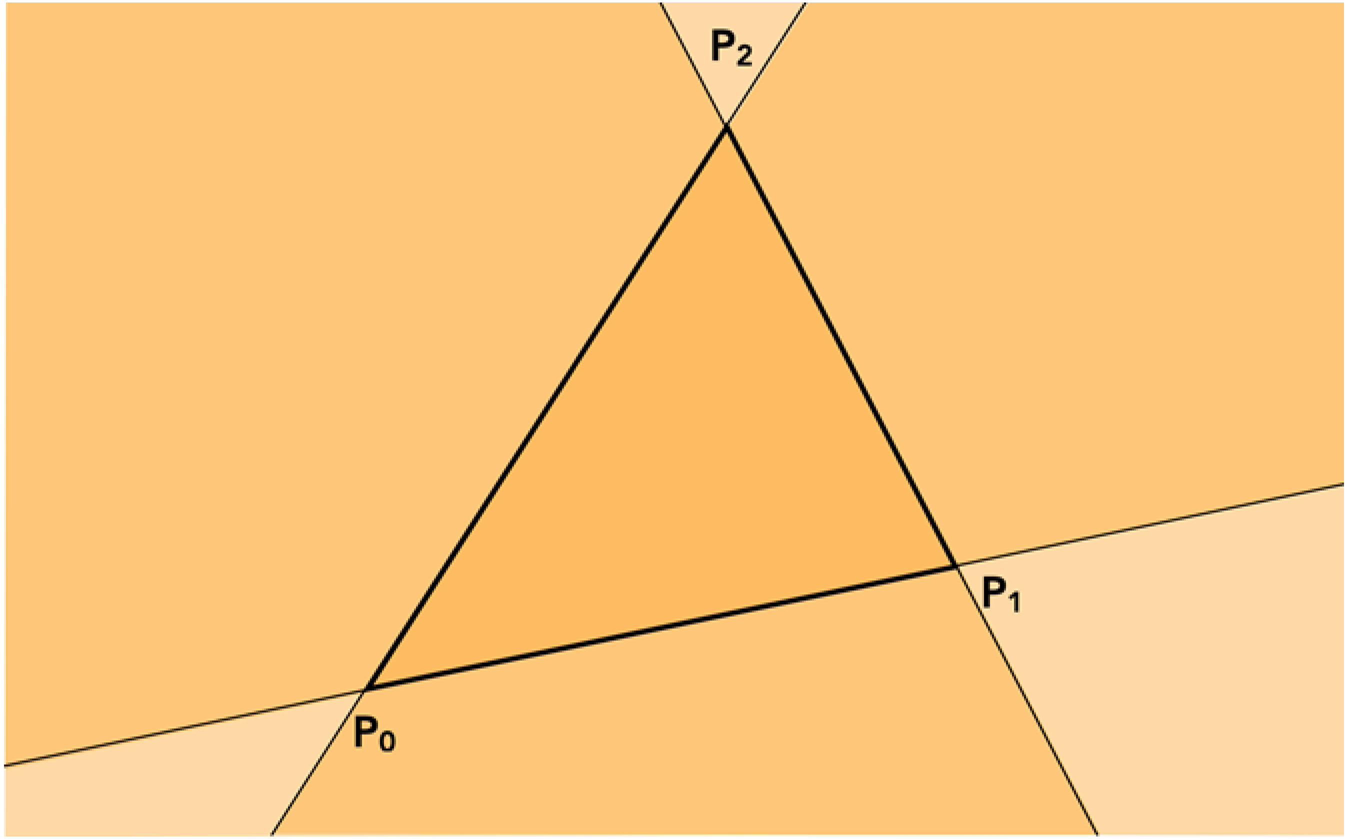
Triângulo

Toda reta divide o plano em que está contida em dois semiplanos



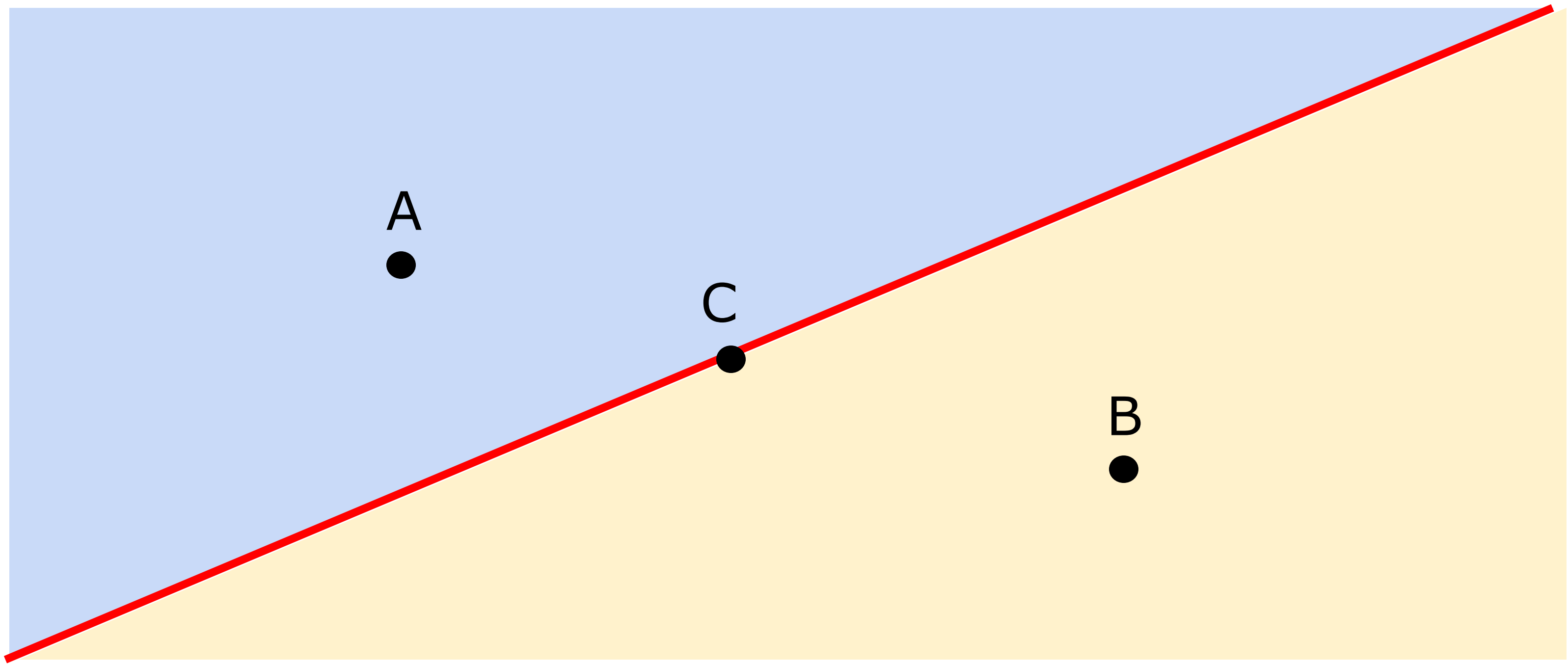
Triângulo

Podemos entender um triângulo como a interseção de três semiplanos



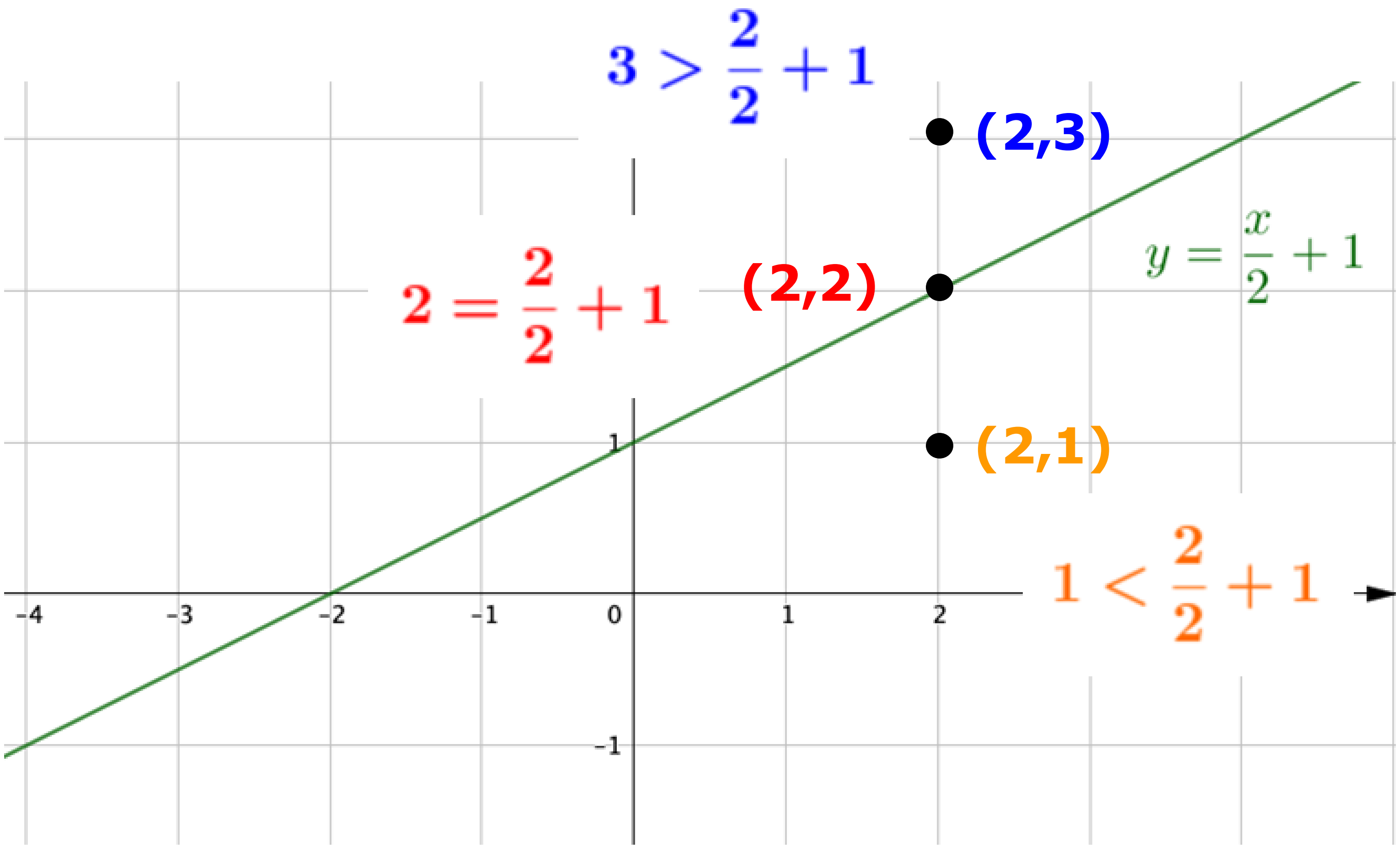
Triângulo

Como decidir em qual semiplano se encontra um ponto dado?



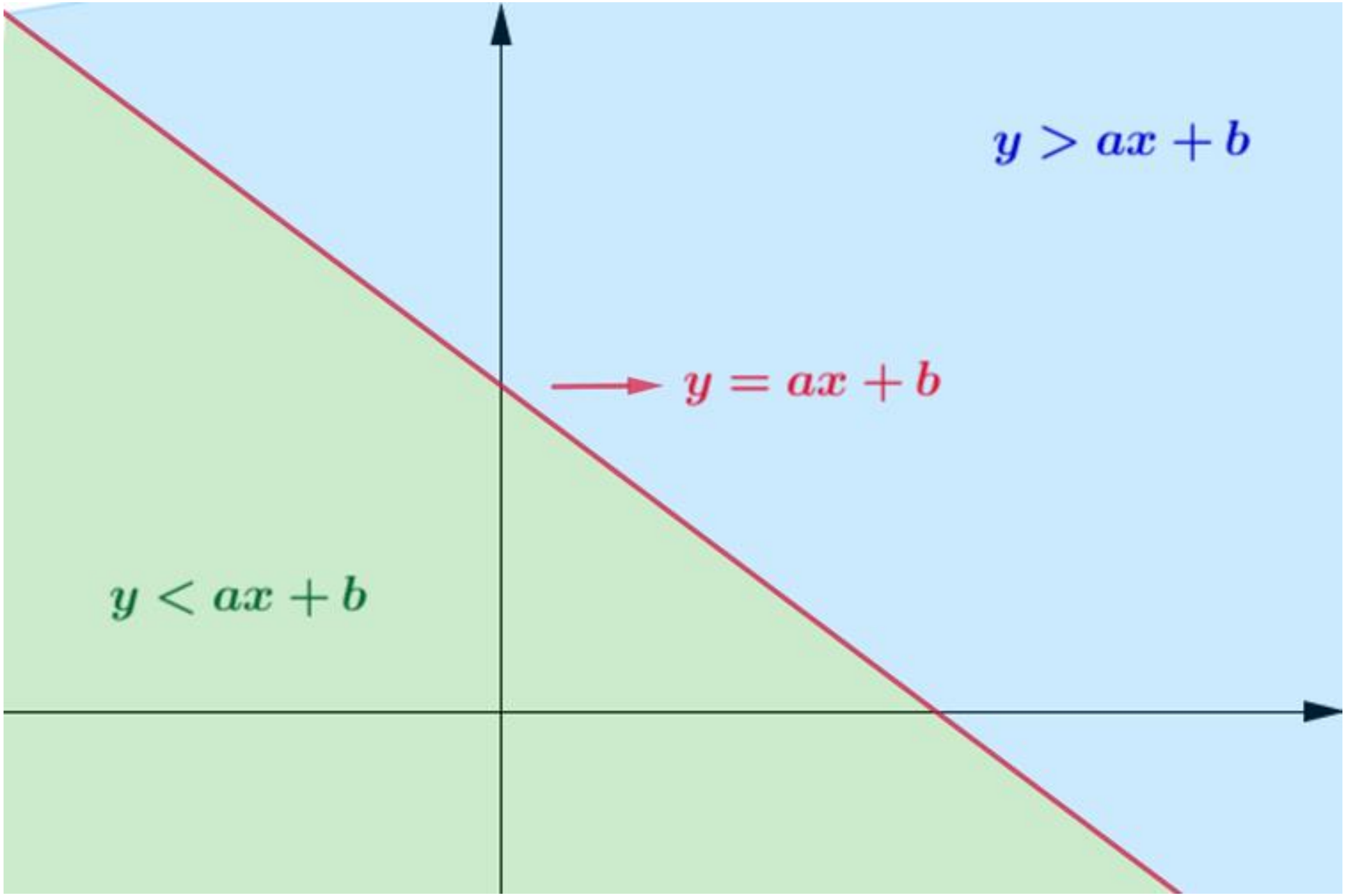
Insper

Geometria Analítica - Ensino Médio



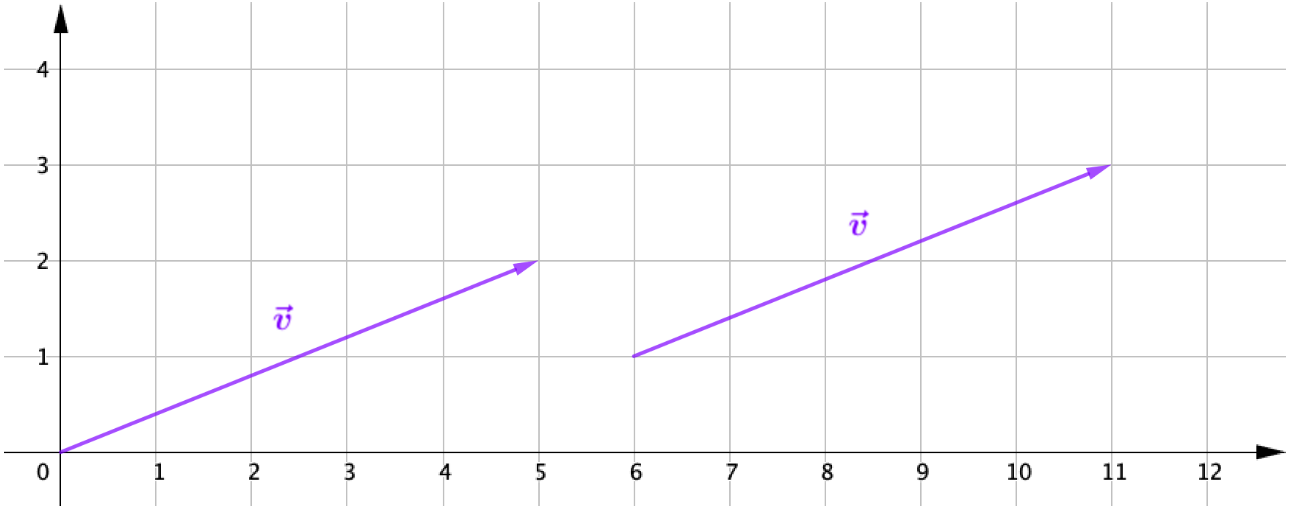
Insper

Geometria Analítica - Ensino Médio



Vetores e a Reta

Neste curso, será fundamental lidar com vetores. Vamos, então, fazer uma abordagem vetorial para nosso problema.

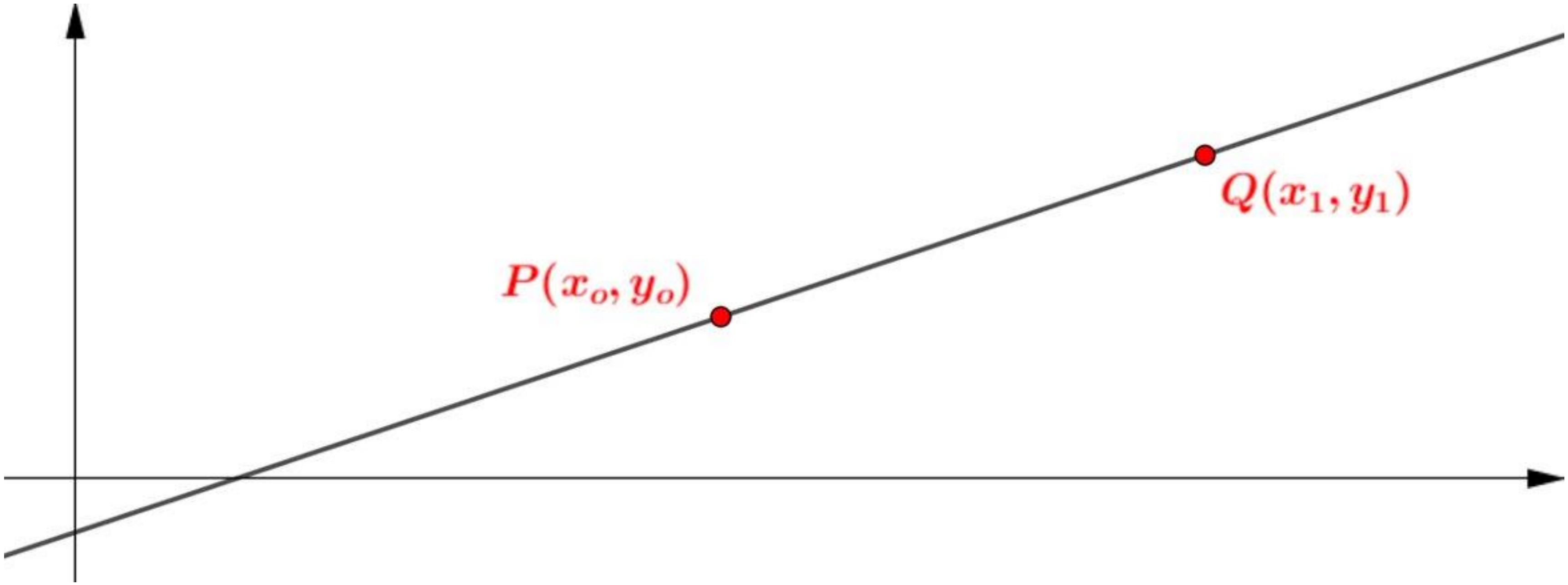


$$\vec{v} = 5\hat{\imath} + 2\hat{\jmath}$$

$$\vec{v} = (5,2)$$

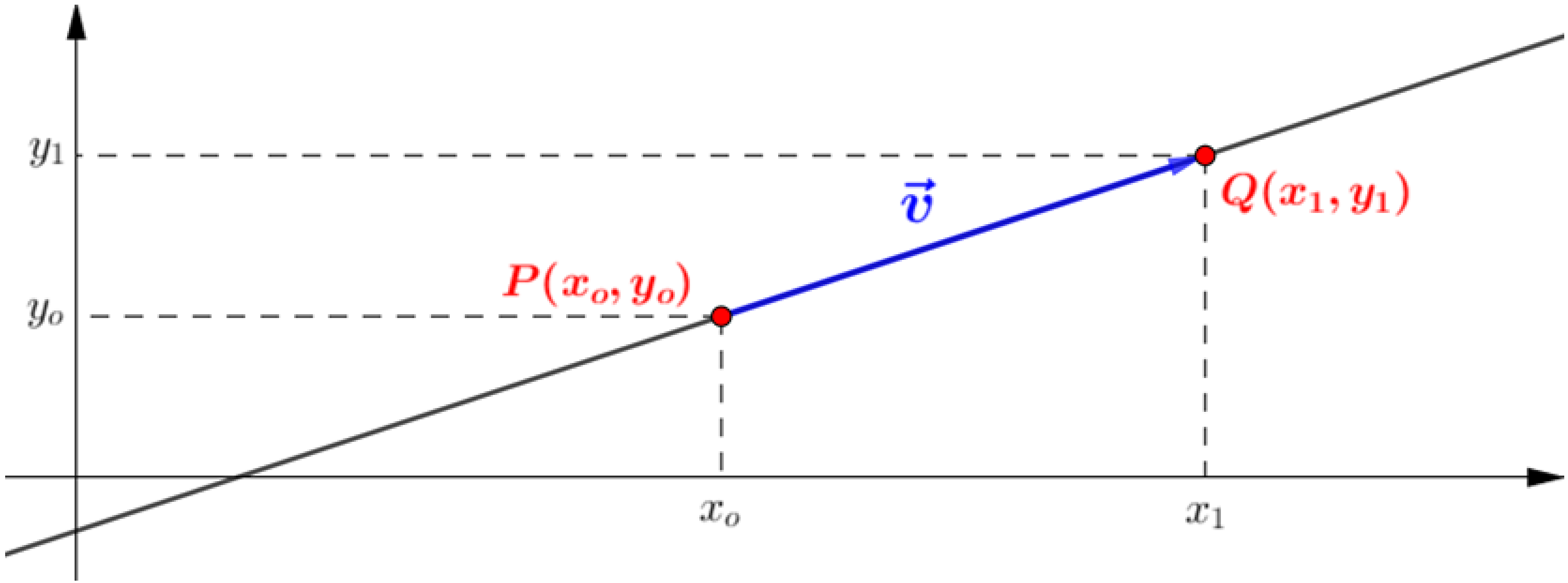
Vetores e a Reta

Vetor diretor de uma reta



Vetores e a Reta

Vetor diretor de uma reta



$$\vec{v} = (x_1 - x_0,\ y_1 - y_o)$$

Insper

Recordando o produto escalar

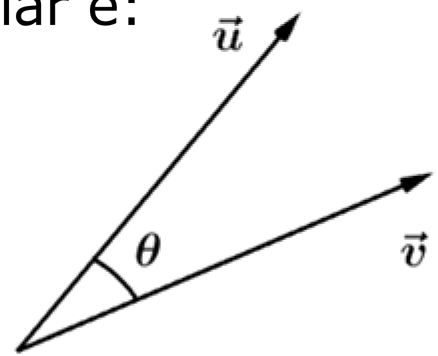
Todos se lembram disso?

O **produto escalar** (ou interno) entre dois vetores $\vec{u} = (u_1, u_2, u_3)$ e $\vec{v} = (v_1, v_2, v_3)$ é:

$$\boxed{\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2 + u_3 v_3}$$

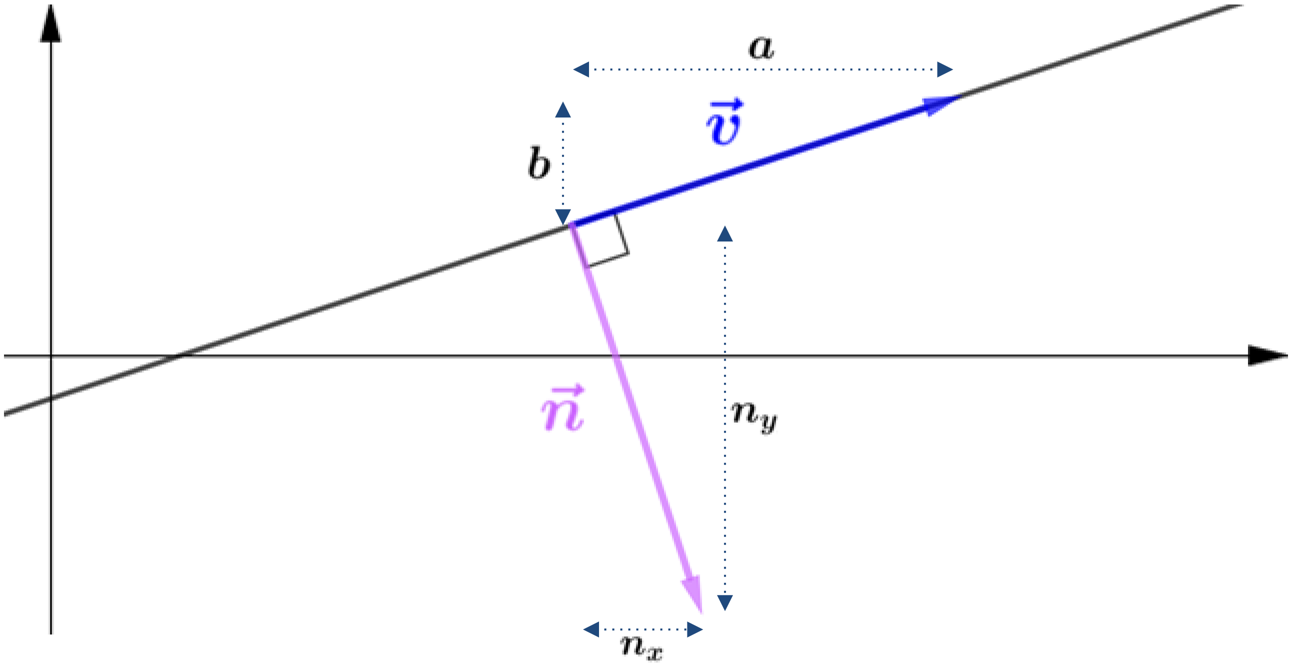
Podemos também falar que o produto escalar é:

$$\boxed{\vec{u} \cdot \vec{v} = ||\vec{u}||\ ||\vec{v}||\ \cos\theta}$$



Insper

Vetor normal à reta no plano



$$\vec{v} \cdot \vec{n} = 0$$

$$(a, b) \cdot (n_x, n_y) = 0$$

Escolhendo n_x = b:

$$(a, b) \cdot (b, n_y) = 0$$

$$ab + b \cdot n_y = 0$$

$$n_y = -a$$

Então:

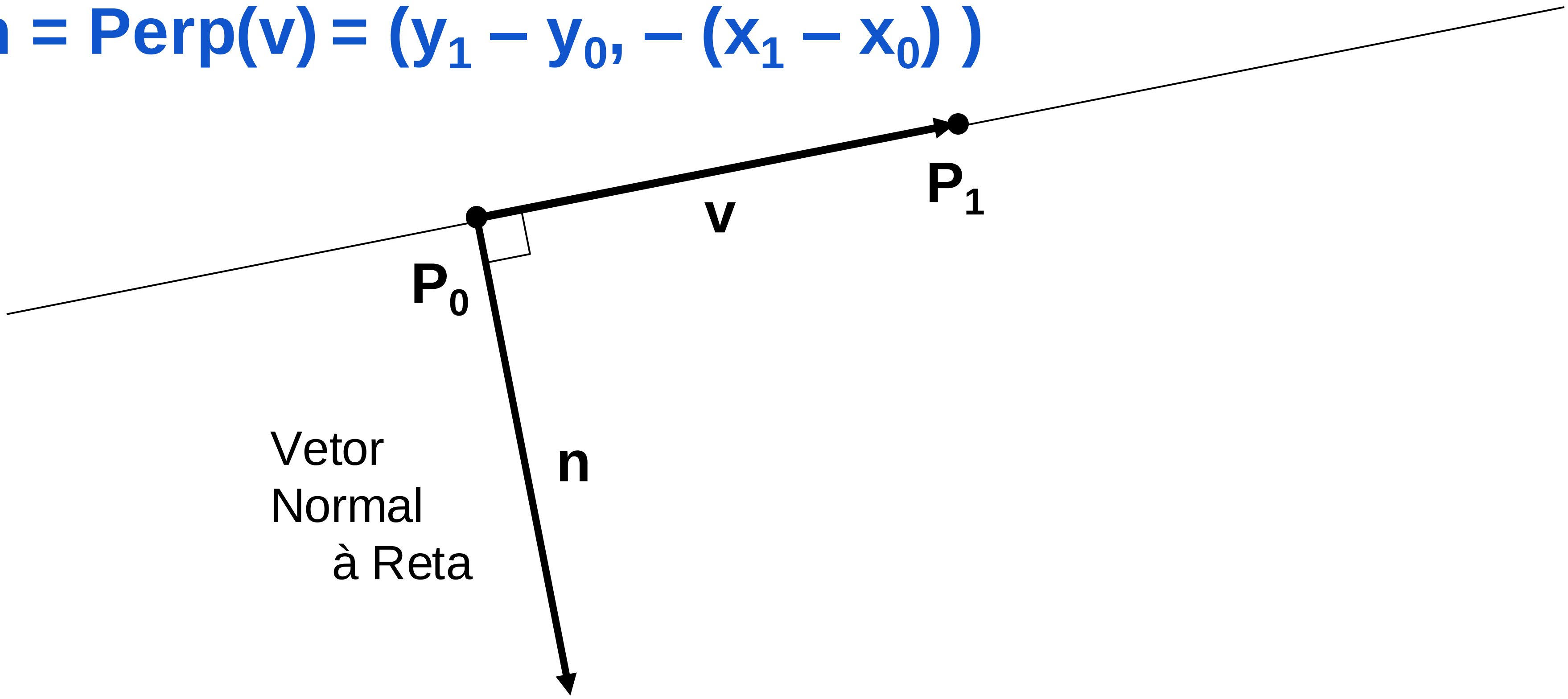
$$\vec{n} = (b, -a)$$

Insper

Trabalhando com o Vetor Normal

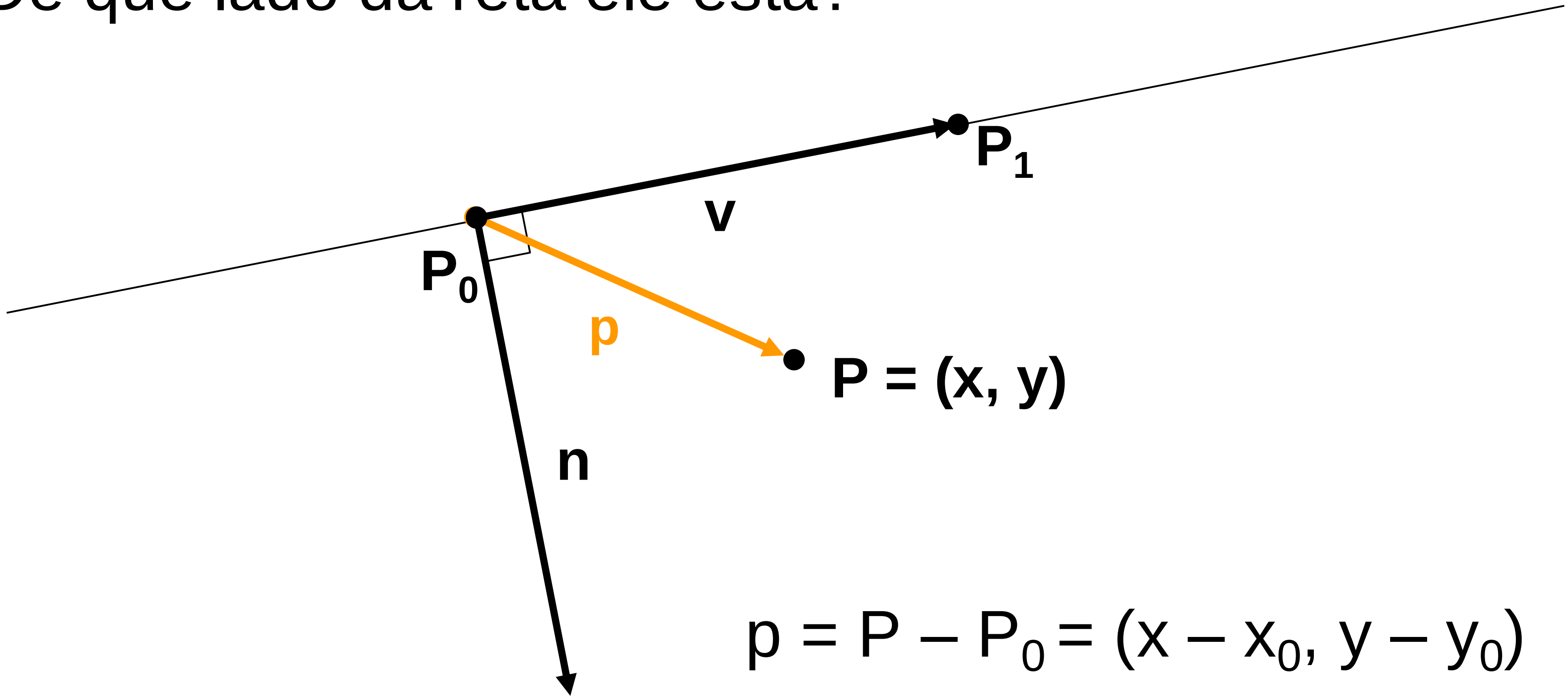
$v = (x_1 - x_0, y_1 - y_0)$

$$\mathbf{n = Perp(v) = (y_1 - y_0, -(x_1 - x_0))}$$



Trabalhando com o Vetor Normal

Considere um ponto qualquer do plano, P = (x, y).
De que lado da reta ele está?

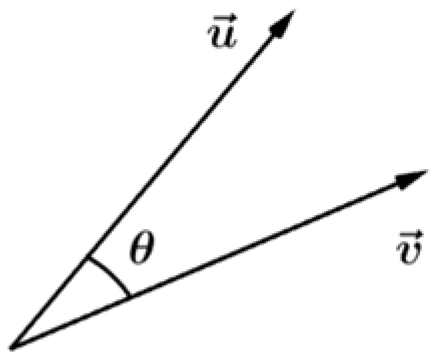


$$p = P - P_0 = (x - x_0, y - y_0)$$

Como já dissemos...

Os vetores $\vec{u}$ e $\vec{v}$ são ortogonais ($\theta = \pi/2$) se, e somente se, o produto escalar entre eles é zero:

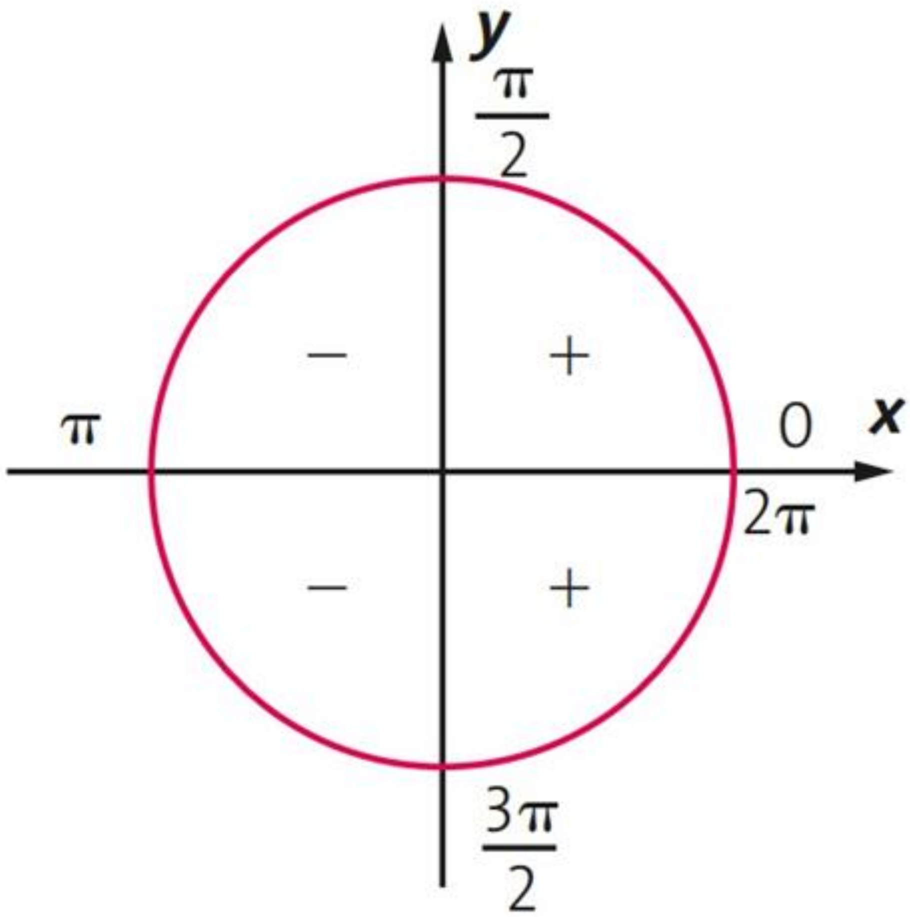
$$\vec{u} \perp \vec{v} \Leftrightarrow \vec{u} \cdot \vec{v} = 0$$



E se for positivo ou negativo?

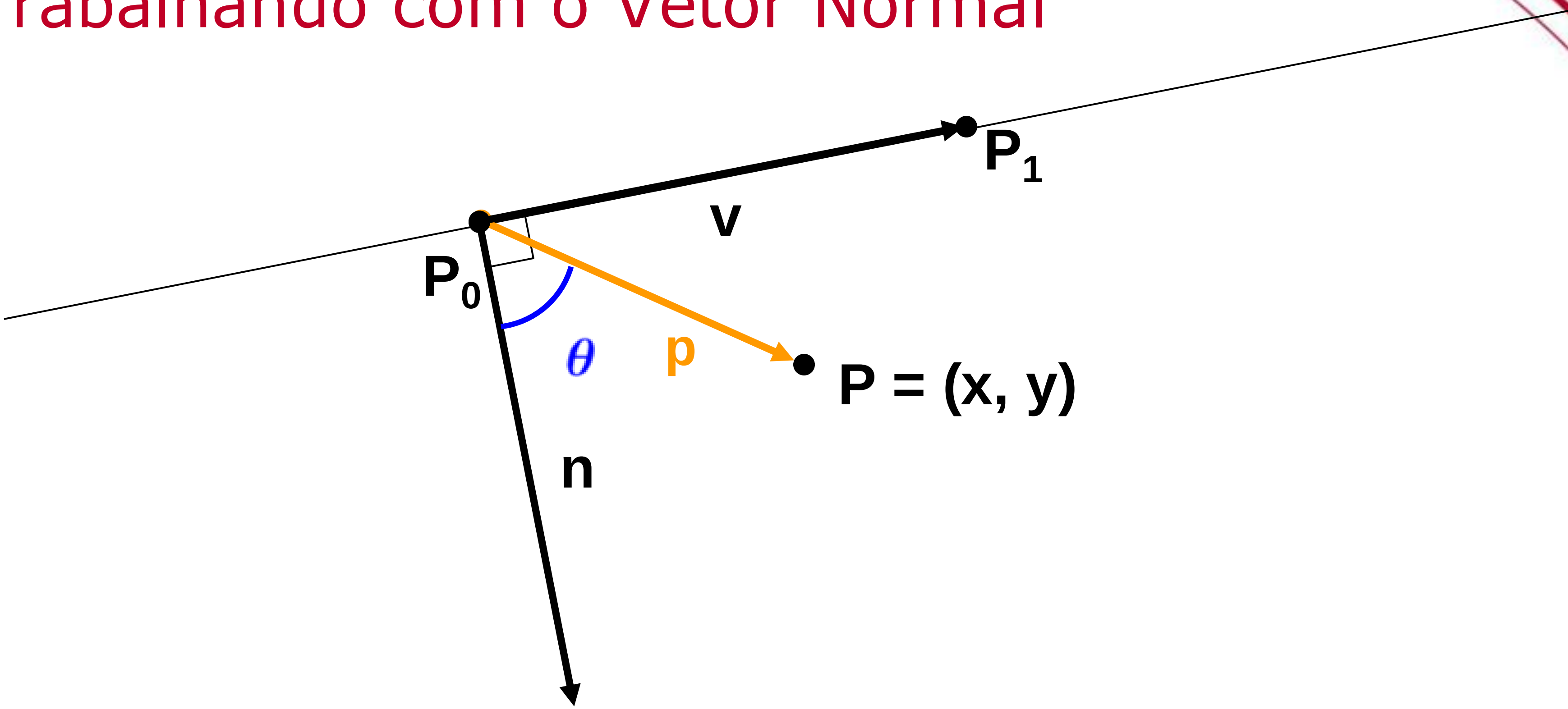
$$\boxed{\vec{u} \cdot \vec{v} = ||\vec{u}|| \; ||\vec{v}|| \; \cos\theta}$$

Cosseno



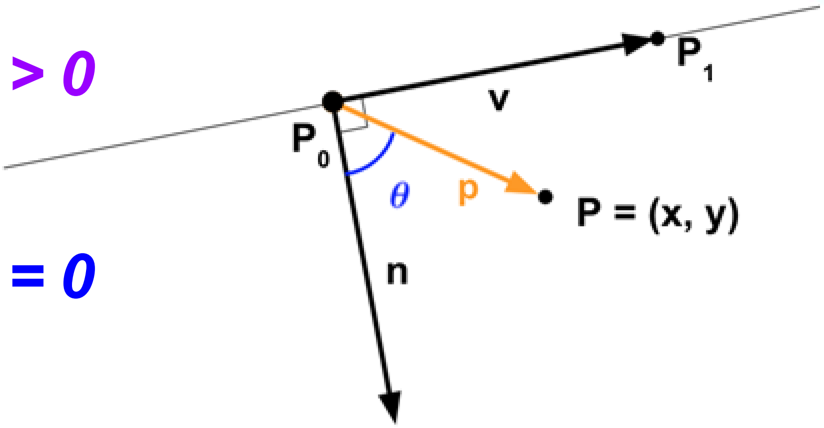
Insper

Trabalhando com o Vetor Normal



- Se $0 \leq \theta < \pi/2$, P está no mesmo semiplano que n.
- Se $\theta = \pi/2$, P pertence à reta.
- Se $\pi/2 < \theta \leq \pi$, P e n estão em semiplanos opostos.

Trabalhando com o Vetor Normal



- Se $0 \le \theta < \pi/2$, P está no mesmo semiplano que n. — $p \bullet n > 0$
- Se $\theta = \pi/2$, P pertence à reta. — $p \bullet n = 0$
- Se $\pi/2 < \theta \le \pi$, P e n estão em semiplanos opostos. — $p \bullet n < 0$

$$L(x, y) = p \bullet n = (x - x_0 \,;\, y - y_0) \bullet (y_1 - y_0 \,;\, -(x_1 - x_0))$$

$$= (x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0)$$

$$= (y_1 - y_0)x - (x_1 - x_0)y + y_0(x_1 - x_0) - x_0(y_1 - y_0)$$

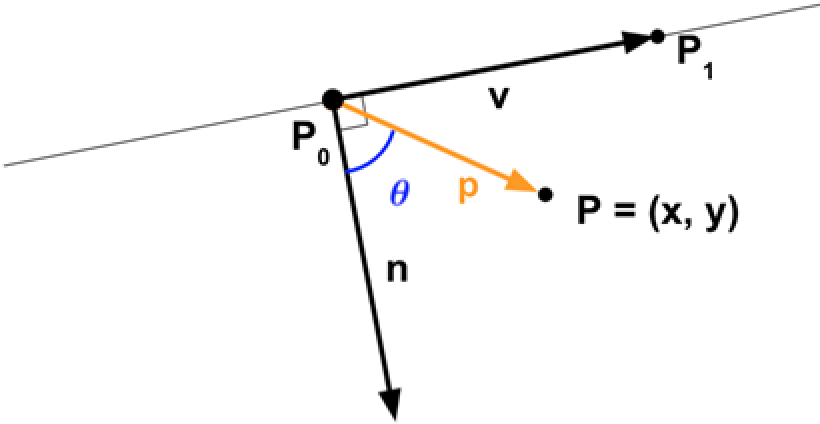
Trabalhando com o Vetor Normal

$$L(x, y) = p \bullet n = (x - x_0\,;\, y - y_0) \bullet (y_1 - y_0\,;\, -(x_1 - x_0))$$

$$= (x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0)$$

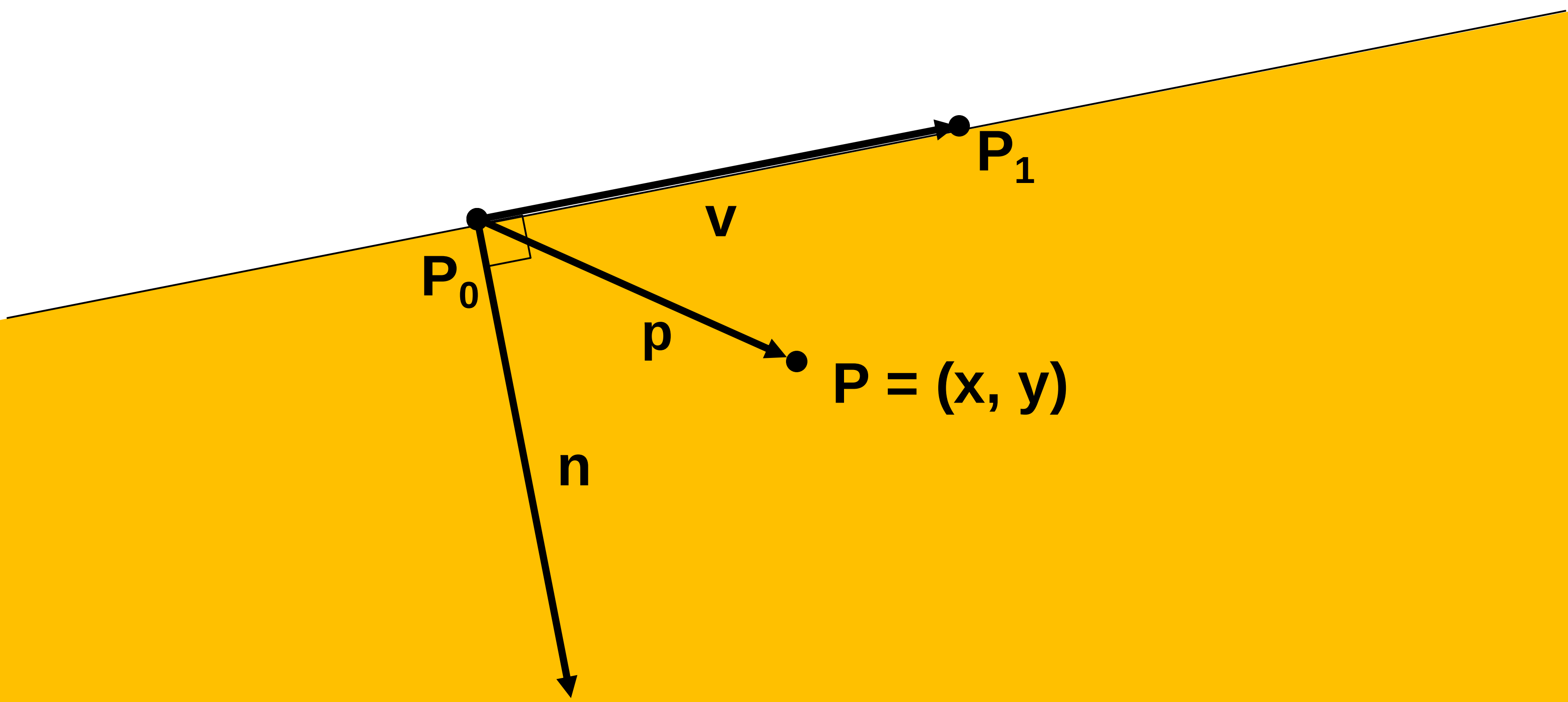
$$= (y_1 - y_0)x - (x_1 - x_0)y + y_0(x_1 - x_0) - x_0(y_1 - y_0)$$

$$L(x, y) = p \bullet n = \mathbf{Ax + By + C}$$



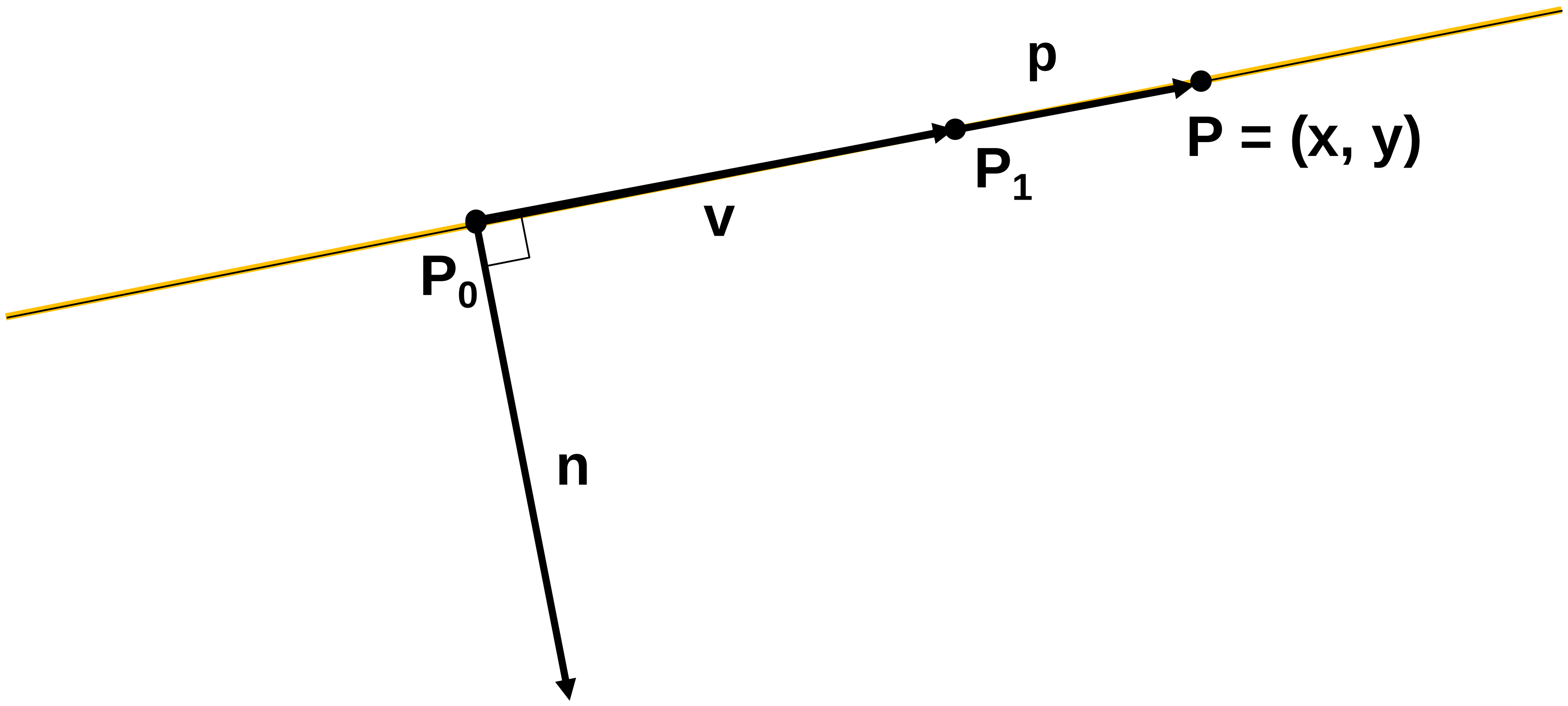
Testando o Ponto na Equação

$$L(x, y) = p \bullet n > 0$$



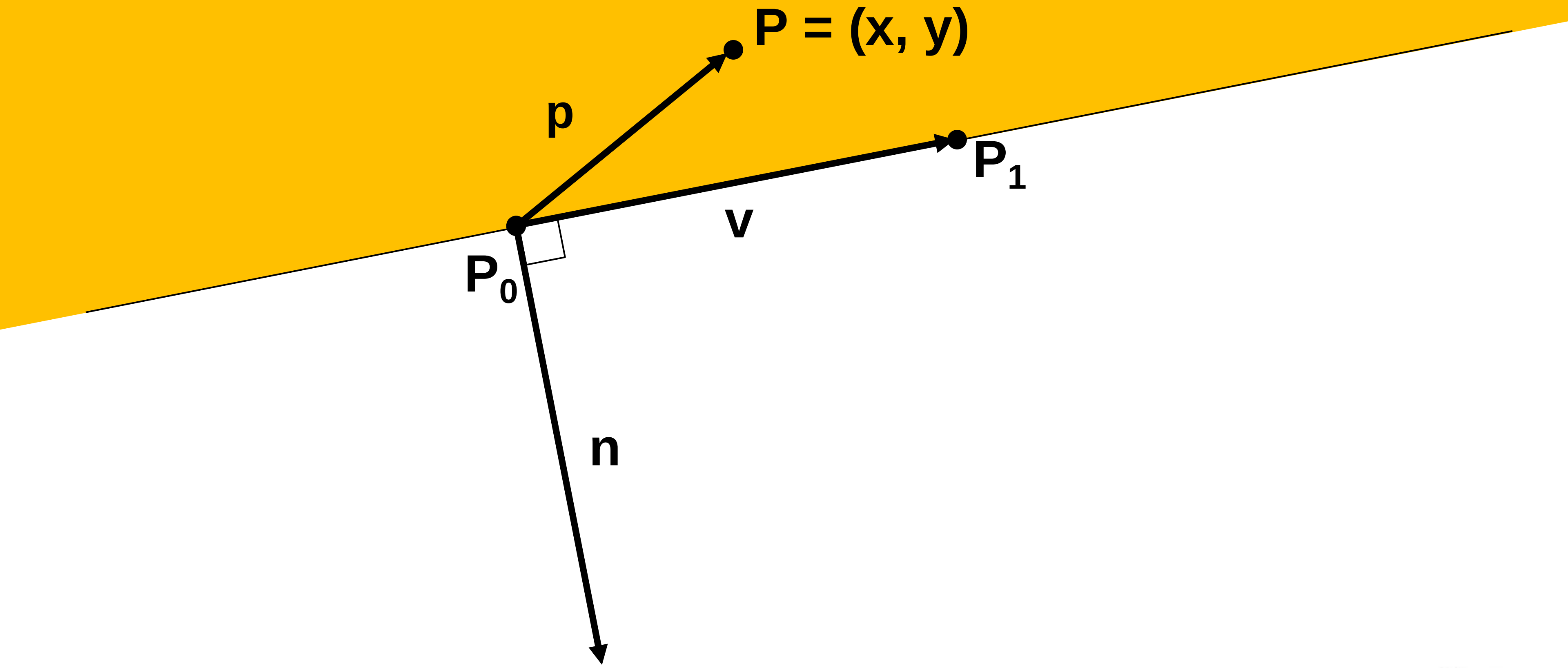
Testando o Ponto na Equação

$$L(x, y) = p \bullet n = 0$$



Testando o Ponto na Equação

$$L(x, y) = p \bullet n < 0$$



Insper

Alternativa para organizar o teste de borda

Essa equação também pode ser organizada como uma matriz:

$$\begin{bmatrix} (x - x_0) & (y - y_0) \\ (x_1 - x_0) & (y_1 - y_0) \end{bmatrix}$$

Se definirmos A = (P – P_0) e B = (P_1 – P_0)

$$\begin{bmatrix} A_x & A_y \\ B_x & B_y \end{bmatrix}$$

O determinante dessa matriz é:

$$A_x * B_y - A_y * B_x$$

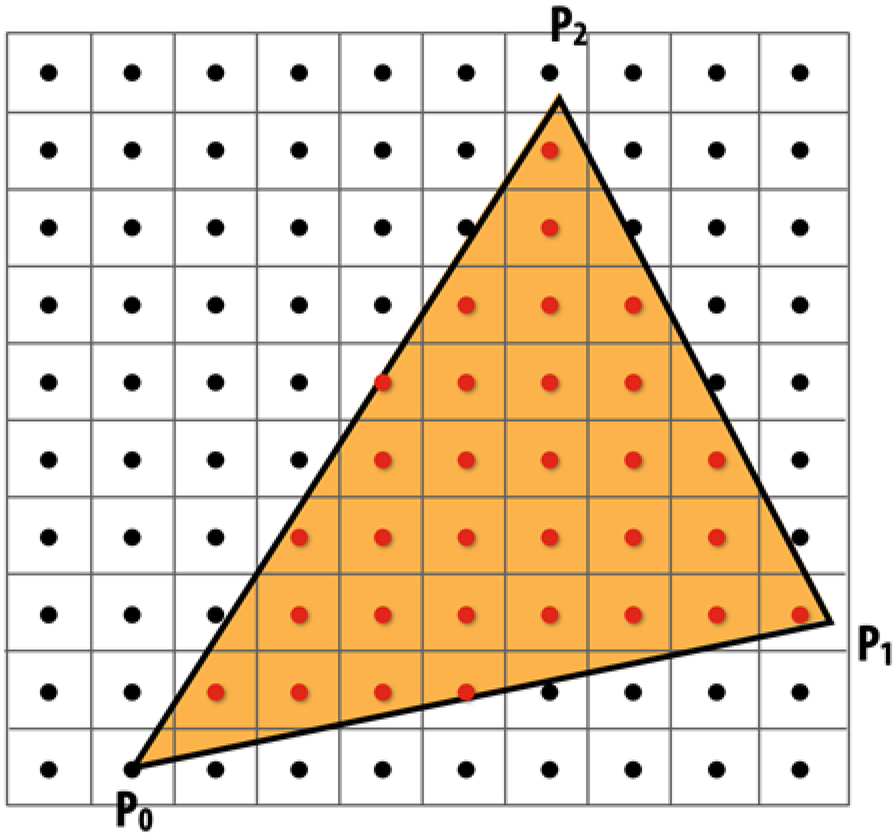
Insper

Teste do ponto no triângulo

O ponto amostrado s = (sx, sy) está dentro do triângulo se estiver "dentro" em relação aos três lados.

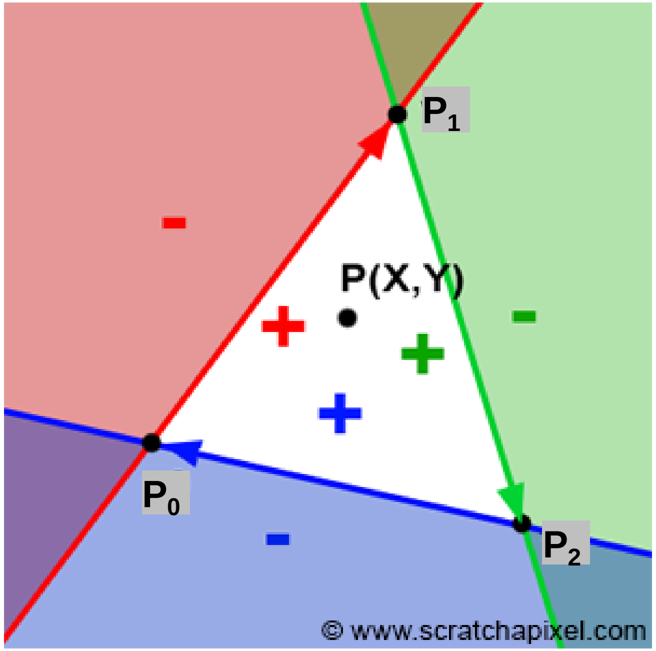
$$inside(sx, sy) = \\ L_0(sx, sy) < 0 \ \&\& \\ L_1(sx, sy) < 0 \ \&\& \\ L_2(sx, sy) < 0;$$

Nota: na prática não são feitos tantos testes assim, existem várias otimizações



Se todos os 3 testes forem positivos

Estamos dentro do triângulo



Esse teste é baseado na estratégia conhecida como Edge Function, proposto por Juan Pineda em 1988 no paper "**A Parallel Algorithm for Polygon Rasterization**".

Atividade

Identifique se o pixel está dentro ou fora do triângulo realizando os cálculos manualmente.

Triângulo: $P_0 = (2,11)$ $P_1=(11,7)$ $P_2=(6, 2)$

Pixels: A = (5, 7) B=(9, 5) C=(8, 13)

$$L(x, y) = (y_1 - y_0)x - (x_1 - x_0)y + y_0(x_1 - x_0) - x_0(y_1 - y_0)$$

$$L(x, y) = (x - x0)(y1 - y0) - (y - y0)(x1 - x0)$$

Se definirmos $A = (P - P_0)$ e $B = (P_1 - P_0)$

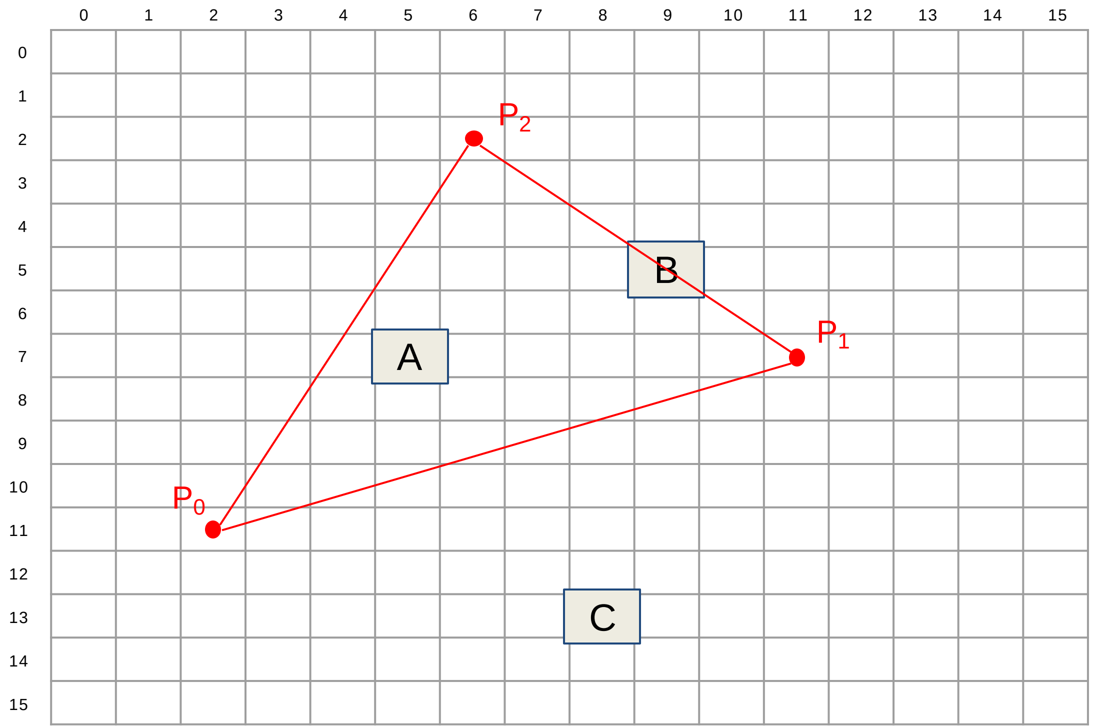
$$\begin{bmatrix} A.x & A.y \\ B.x & B.y \end{bmatrix}$$

Obs:

1. considere o eixo y invertido, assim verifique se as equações L(x, y) são positivas.
2. considere que os pontos P_0, P_1 e P_2 estão exatamente no centro do pixel.

Atividade

Identifique se o pixel está dentro ou fora do triângulo realizando os cálculos manualmente.



Insper

Resolvendo

$P_0 = (2,11)$ $P_1=(11,7)$ $P_2=(6, 2)$

$A = (5, 7)$ $B=(9, 5)$ $C=(8, 13)$

$$\boldsymbol{L(x, y) = (y_1 - y_0)x - (x_1 - x_0)y + y_0(x_1 - x_0) - x_0(y_1 - y_0)}$$

$L_1(x, y) = (7 - 11)x - (11 - 2)y + 11(11 - 2) - 2(7 - 11)$
$L_1(x, y) = -4x - 9y + 107$

$L_2(x, y) = (2 - 7)x - (6 - 11)y + 7(6 - 11) - 11(2 - 7)$
$L_2(x, y) = -5x + 5y + 20$

$L_3(x, y) = (11 - 2)x - (2 - 6)y + 2(2 - 6) - 6(11 - 2)$
$L_3(x, y) = 9x + 4y - 62$

$A = (5, 7)$ $B=(9, 5)$ $C=(8, 13)$

$L_1(x, y) = -4x - 9y + 107$

$L_2(x, y) = -5x + 5y + 20$

$L_3(x, y) = 9x + 4y - 62$

A)

inside(5, 7) = $L_1(5, 7) \geq 0$ && $L_2(5, 7) \geq 0$ && $L_3(5, 7) \geq 0$

inside(5, 7) = $-4*5 - 9*7 + 107 \geq 0$ && $-5*5 + 5*7 + 20 \geq 0$ && $9*5 + 4*7 - 62 \geq 0$

inside(5, 7) = $-20 - 63 + 107 \geq 0$ && $-25 + 35 + 20 \geq 0$ && $45 + 28 - 62 \geq 0$

inside(5, 7) = $24 \geq 0$ && $30 \geq 0$ && $11 \geq 0$

inside(5, 7) = True (dentro)

B)

inside(9, 5) = $L_1(9, 5) \geq 0$ && $L_2(9, 5) \geq 0$ && $L_3(9, 5) \geq 0$

inside(9, 5) = $-4*9 - 9*5 + 107 \geq 0$ && $-5*9 + 5*5 + 20 \geq 0$ && $9*9 + 4*5 - 62 \geq 0$

inside(9, 5) = $-36 - 45 + 107 \geq 0$ && $-45 + 25 + 20 \geq 0$ && $81 + 20 - 62 \geq 0$

inside(9, 5) = $26 \geq 0$ && $0 \geq 0$ && $39 \geq 0$

inside(9, 5) = True (dentro, intersectando um dos lados)

A = (5, 7) B=(9, 5) C=(8, 13)

$L_1(x, y) = -4x - 9y + 107$

$L_2(x, y) = -5x + 5y + 20$

$L_3(x, y) = 9x + 4y - 62$

C)

inside(8, 13) = $L_1(8, 13) \geq 0$ && $L_2(8, 13) \geq 0$ && $L_3(8, 13) \geq 0$

inside(8, 13) = $-4*8 - 9*13 + 107 \geq 0$ && $-5*8 + 5*13 + 20 \geq 0$ && $9*8 + 4*13 - 62 \geq 0$

inside(8, 13) = $-32 - 117 + 107 \geq 0$ && $-40 + 65 + 20 \geq 0$ && $72 + 52 - 62 \geq 0$

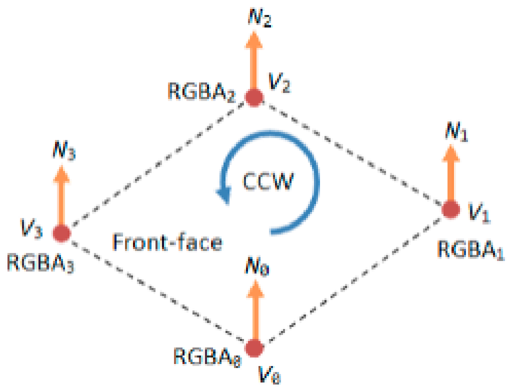
inside(8, 13) = $-42 \geq 0$ && $45 \geq 0$ && $62 \geq 0$

inside(8, 13) = False (fora)

Ordem dos Pontos

Perceba que a ordem dos pontos é importante, senão você poderá estar pegando semiplanos errados.

Ao se criar um polígono em OpenGL, a ordem padrão para se conectar os vértices é no sentido anti-horário. Isso é conhecido como a "winding order".



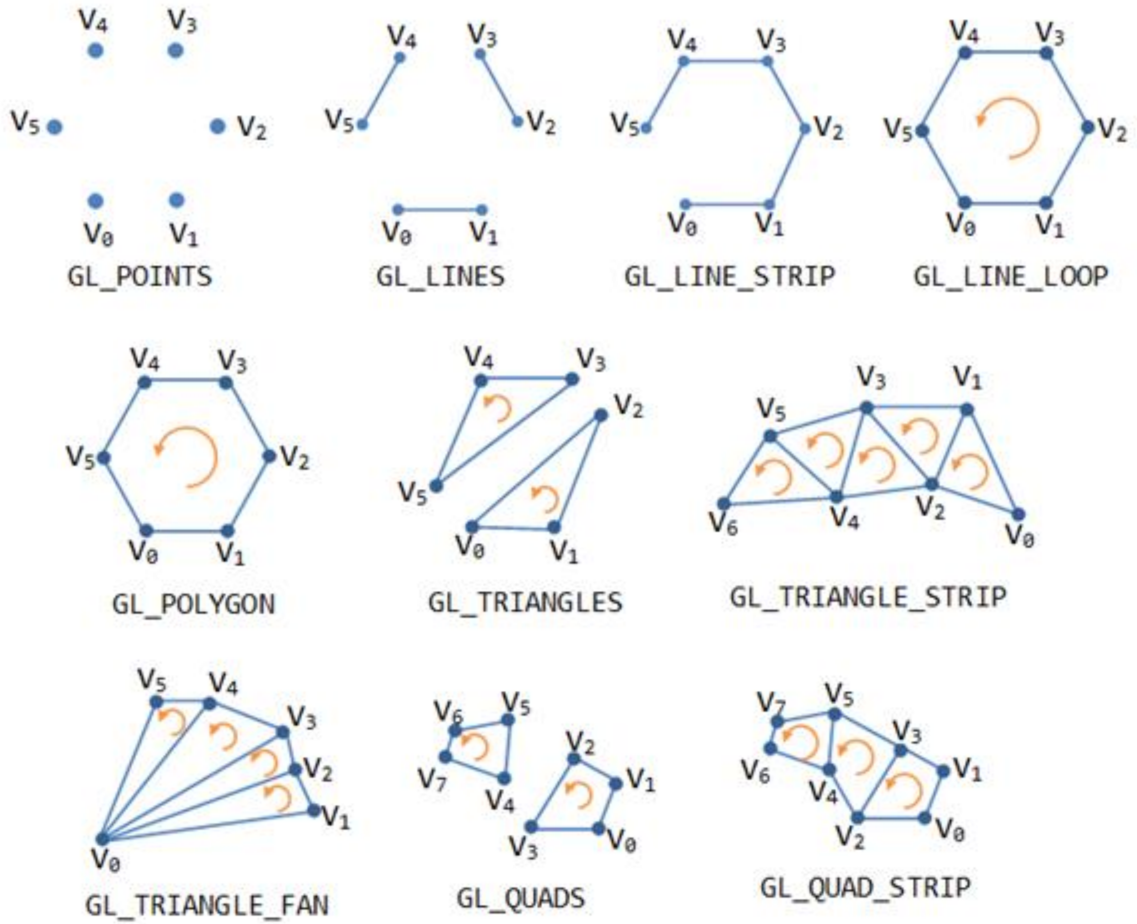
A estratégia de ordem pode ser alterada invocando a função:

void glFrontFace(GLenum *mode*);

o parâmetro pode ser: GL_CW (horário) ou GL_CCW (anti-horário)

Insper

Primitivas Gráficas em OpenGL



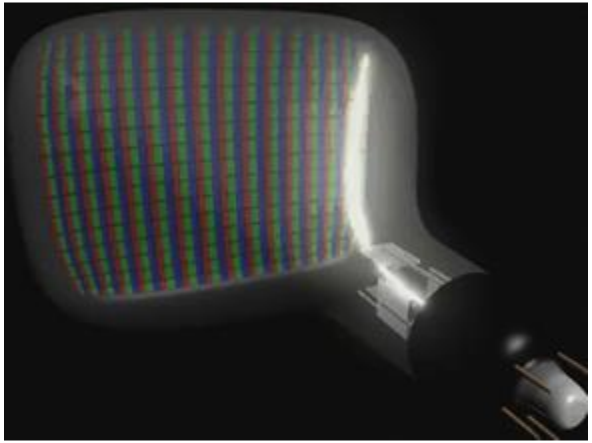
https://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html

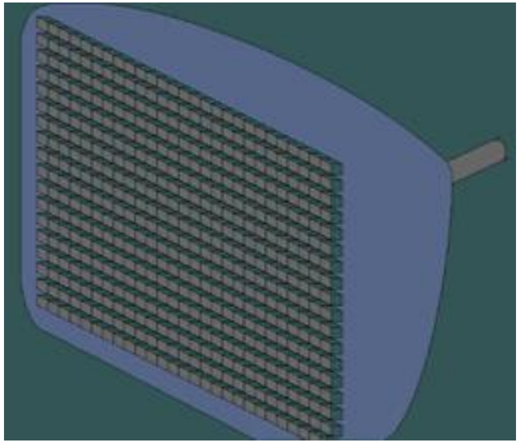
Insper

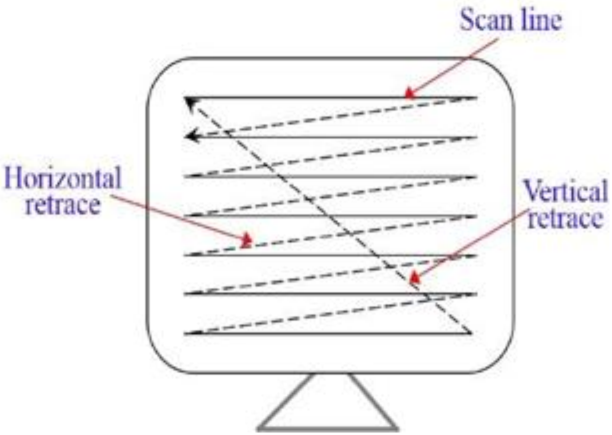
Eixos das Imagens

"However, in computer graphics and image processing one often uses a coordinate system with the y axis pointing down (as displayed on the computer's screen). This convention developed in the 1960s (or earlier) from the way that images were originally stored in display buffers."

Fonte: https://simple.wikipedia.org/wiki/User:Jeffwang/Cartesian_coordinate_system



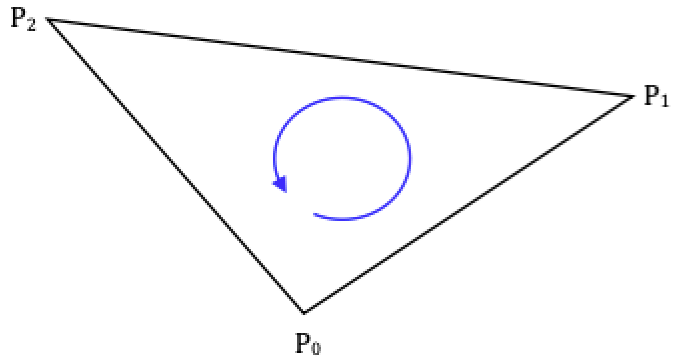




Insper

Cuidado

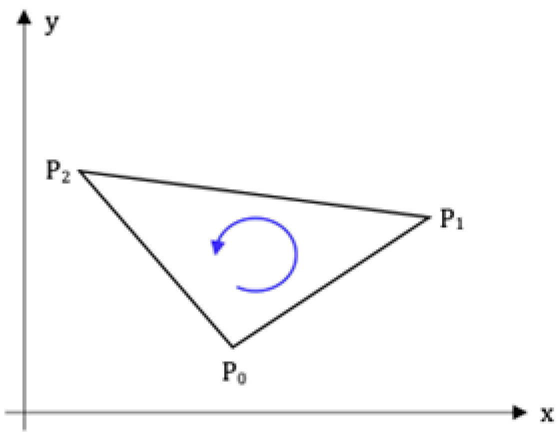
Pessoal, um ponto de atenção sobre as convenções para a ordem dos vértices do triângulo quando queremos decidir se um ponto está dentro ou fora desse triângulo. Convencionamos adotar sempre o sentido anti-horário, como ilustrado na figura abaixo.

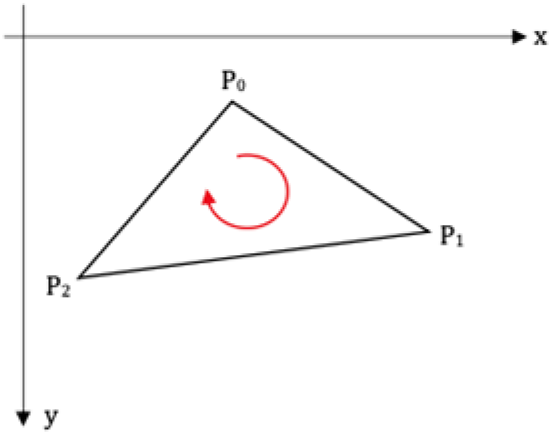


Com isso, podemos afirmar que um ponto pertence ao interior do triângulo quando os valores numéricos das três expressões Li(x, y) são negativos. Chegamos a essa conclusão adotando a orientação convencional do plano cartesiano, ou seja, o eixo x para a direita e o eixo y para cima.

Invertendo

No exercício e também no projeto a ser entregue, usamos o sistema de coordenadas usual do sistema de pixels, ou seja, com o eixo x orientado para a direita, mas com o eixo y orientado para baixo. Observe o que acontece com o mesmo triângulo representado nesses dois sistemas.

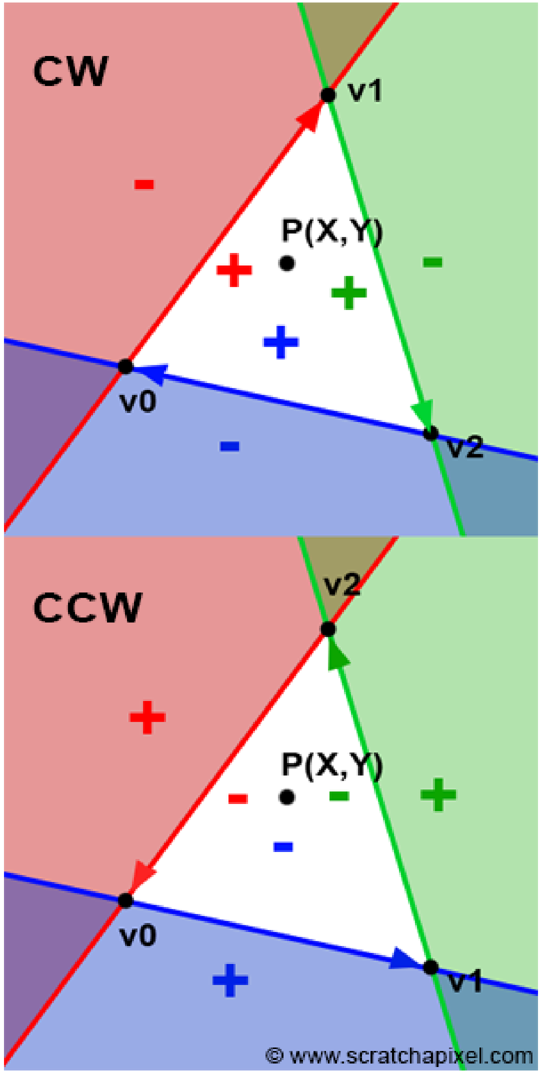




Isso mesmo! O sentido do polígono inverte, ou sejam, fica alterado de anti-horários para horário e vice-versa.

E o que fazer?

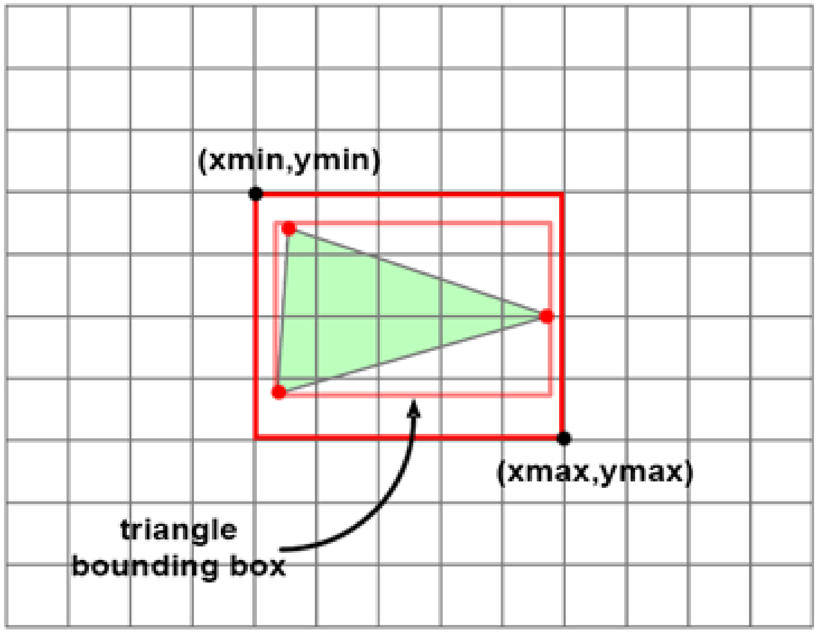
Calma, tudo continua valendo da mesma forma, só que os sinais das três expressões Li(x, y) ficarão invertidos. Então, usando esse sistema de coordenadas com o eixo y apontando para baixo, podemos afirmar que um ponto pertence ao interior do triângulo quando os valores numéricos das três expressões Li(x, y) são positivos. Fiquem atentos a essa questão e bom trabalho!



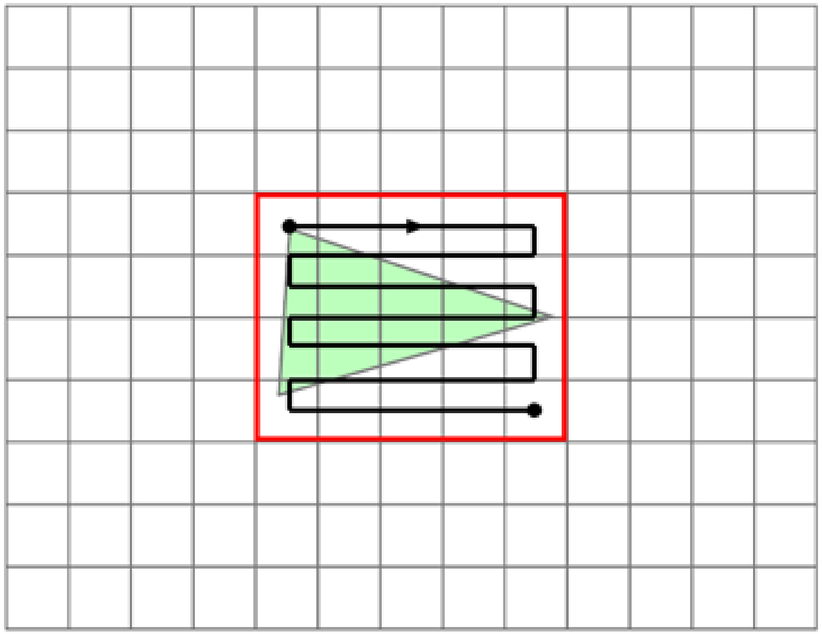
Insper

Otimizações: Bounding Box de Triângulos 2D

Para evitar a iteração em todos os pixels da imagem, podemos iterar em todos os pixels contidos na caixa delimitadora (Bounding Box) do triângulo 2D.



© www.scratchapixel.com



© www.scratchapixel.com

Insper

Traversal de Triângulos

Traversal de triângulos é implementado nas GPUs modernas.
Um artigo descrevendo o funcionamento se encontra em:
http://oa.upm.es/9184/1/INVE_MEM_2010_84947.pdf

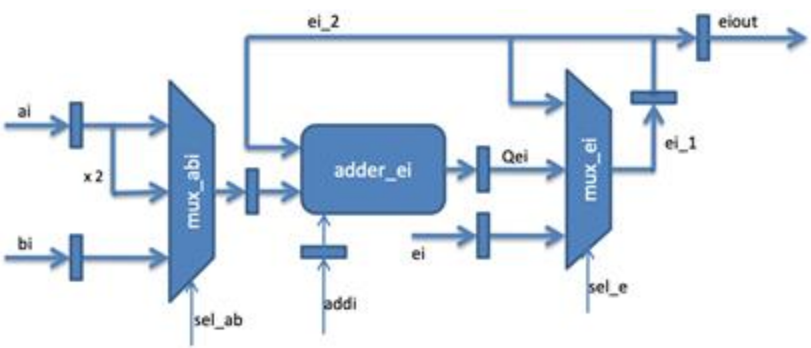


Fig. 6. Full schematic of an e_i adder.

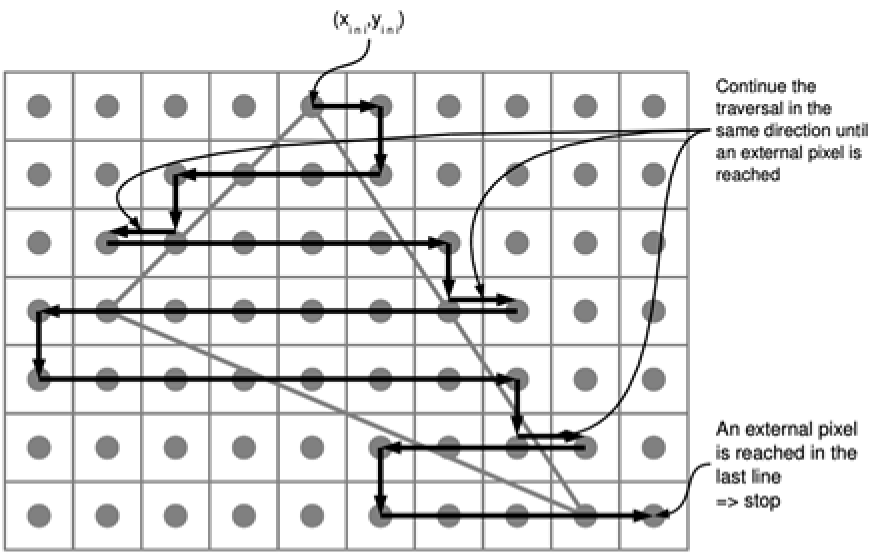


Fig. 7. Zig-zag traversal algorithm.

Referência recomendada

Home

Donate ♥

Rasterization: a Practical Implementation

Distributed under the terms of the CC BY-NC-ND 4.0 License.

https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm.html

Projeto 1 : Primeira Parte

Fazer um renderizador:

1 – Capaz de desenhar Pontos

2 – Capaz de desenhar Linhas

3 – Capaz de desenhar Triângulos

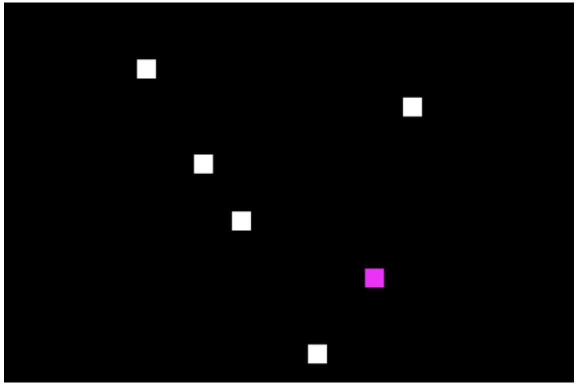
Data de Entrega:

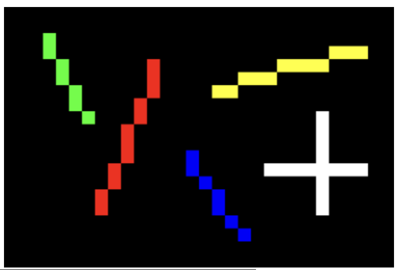
14/8/2024 às 23:59, via Blackboard (https://insper.blackboard.com/)

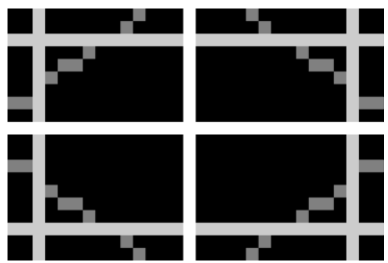
Detalhes:

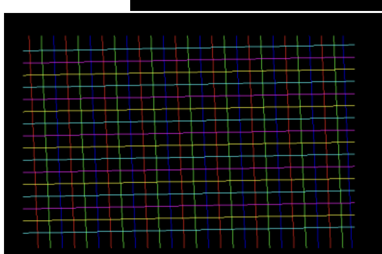
Página da Disciplina (https://lpsoares.github.io/ComputacaoGrafica/)

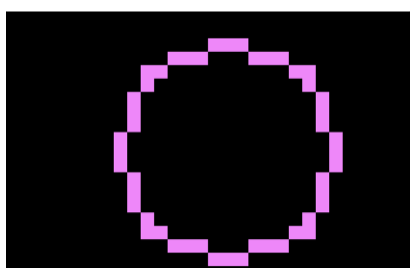
Exemplos

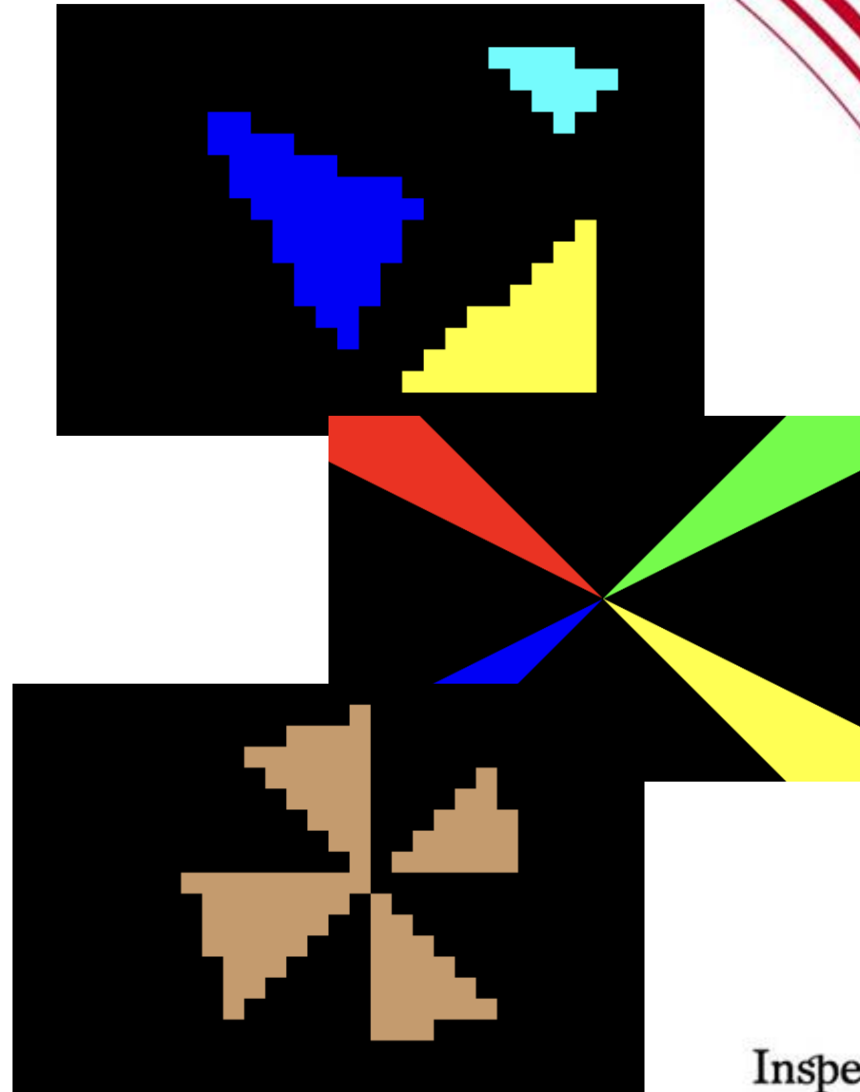






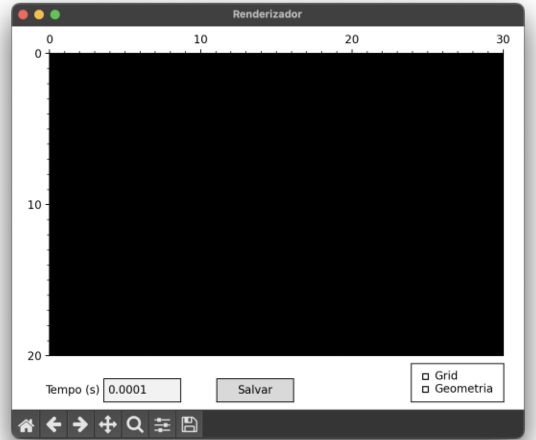


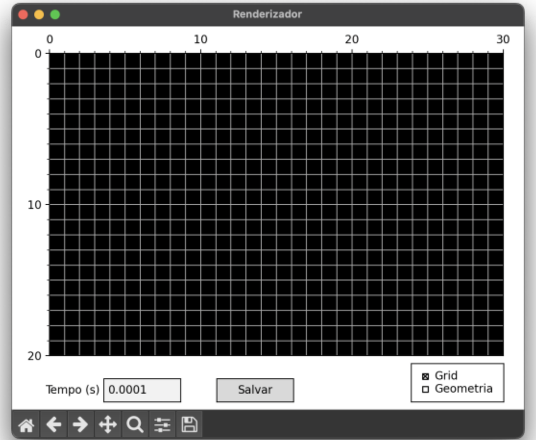


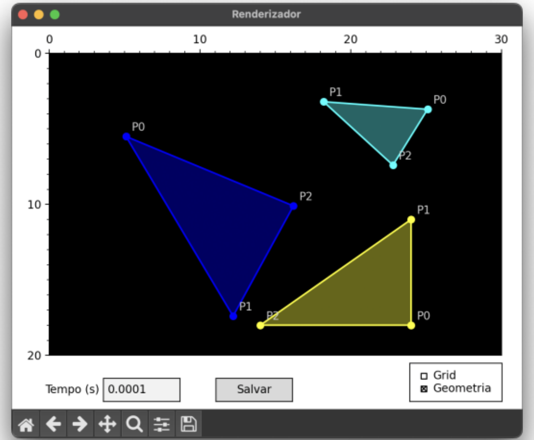


Renderizador

Possível mostrar Grid e Geometria (nessa fase do projeto).



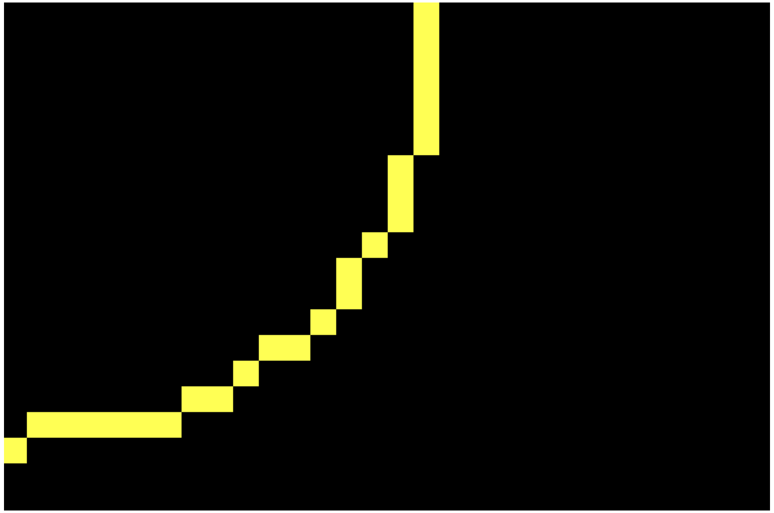




Extra

Desenhar um círculo (ou pelo menos parte dele).

```
<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Circle2D radius='16'/>
        <Appearance>
          <Material emissiveColor='1 1 0'/>
        </Appearance>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```



Insper

Computação Gráfica

Luciano Soares
<lpsoares@insper.edu.br>

Fabio Orfali
<fabioO1@insper.edu.br>