Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Engenharia da Computação

Computação Gráfica

**Rasterização**

Alunos:

José Arlindo Alcântara da Silva

Fernando Weslley Silva de Oliveira

Fortaleza - CE, 30 de Outubro de 2021

**1.Introdução a rasterização de retas**

A rasterização de retas compreende no processo de transformar uma representação vetorial em uma representação matricial. viabilizando assim, que imagens tridimensionais possam ser representadas em forma matricial como é mostrado na figura 1 , onde os pixels selecionados para formar um reta estão destacados em cinza e a reta a ser desenhada está destacada em preto.

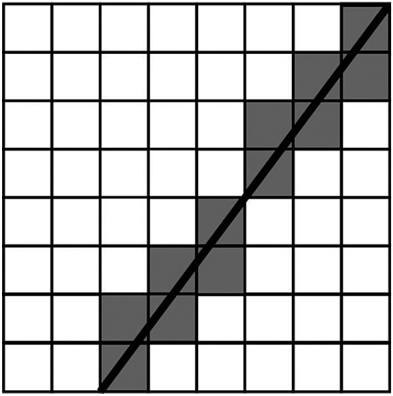


Figura 1.

Fonte: Livro- Computação Gráfica:Teoria e Prática

A partir da análise da Figura 1, fica clara a dificuldade para representação de uma reta em formato raster, já que dependendo da inclinação da reta, fica evidente as deformações geradas na reta devido ao formato dos pixels. Dessa forma, se torna necessário utilizar técnicas auxiliares para minimizar as distorções geradas.

**2.Objetivos**

2.1 Implementar o algoritmo de rasterização de retas, utilizando diferentes resoluções de imagens e variações das retas nos eixos X e Y.

Situações:  **|ΔX | >|ΔY| e |ΔY| > |ΔX |**

Obter e comparar: 4 semirretas diferentes para 3 resoluções diferentes.

Avaliar situação: Reta na vertical ou na horizontal.

.

**3. Resultados da implementação do algoritmo de rasterização de retas.**

O algoritmo de rasterização de retas foi implementado em python e está disponível no seguinte link: ​<https://replit.com/@ArlindoAlcntara/Rasterizacao>. Dessa forma, ​Para realizar a representação de pixels, utilizou-se das grades presentes em gráficos, assim cada quadrado no gráfico representa um pixel e o respectivo ponto marcado representa o centro do pixel. A nível de implementação foi considerado que todo pixel teria tamanho de 1 para qualquer resolução, assim, para aumentar a resolução de uma imagem, é necessário dobrar o tamanho de representação da reta em X e Y para o modelo adotado.

3.1 Mesma reta com diferentes resoluções ​ **ΔX > ΔY**

Para representar um aumento de resolução, foi utilizada uma constante N para representar quantas vezes a resolução seria aumentada, ou seja, mais pixels no mesmo espaço. Com isso, a Figura 2 é a reta original com resolução 1 e as figuras 3 e 4 representam resolução 4 e 8 vezes maior que a Figura 2 utilizando a variação em x de 0 a 19 e em y de 0 a 11.

Ou seja:

(X1, Y1) = (0,0)

(X2,Y2) = (19,11)

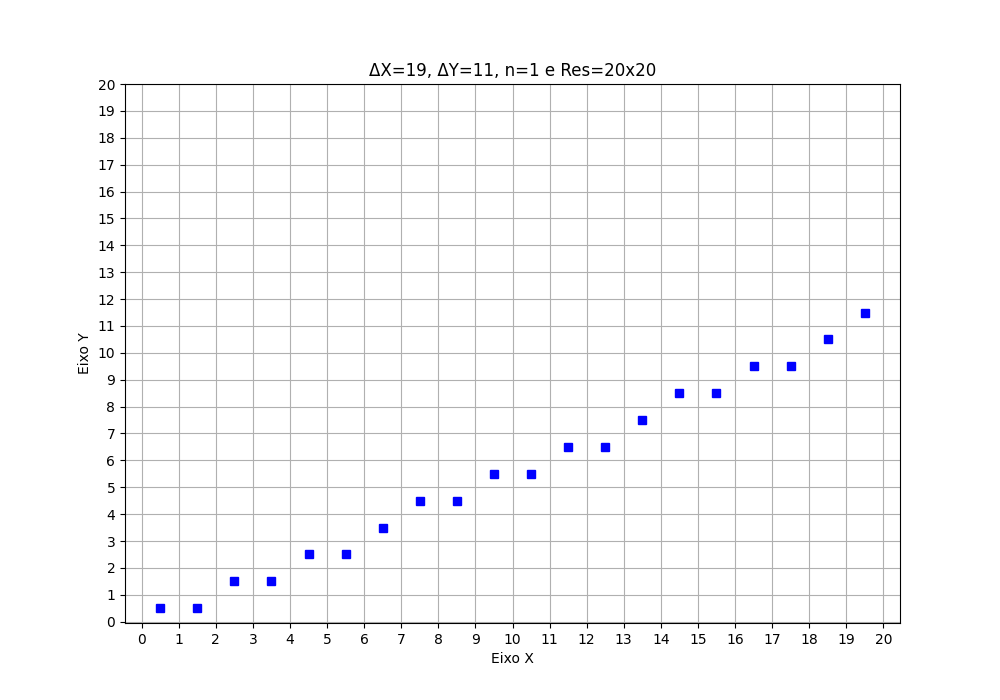


Figura 2. Fonte: Própria da dupla

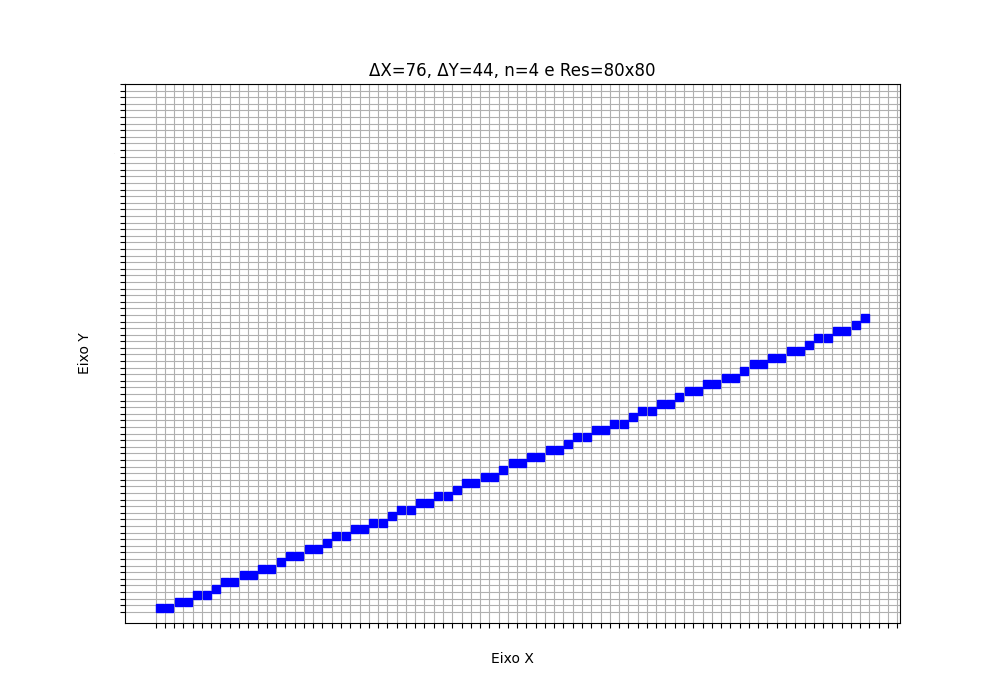


Figura 3. Fonte: Própria da dupla

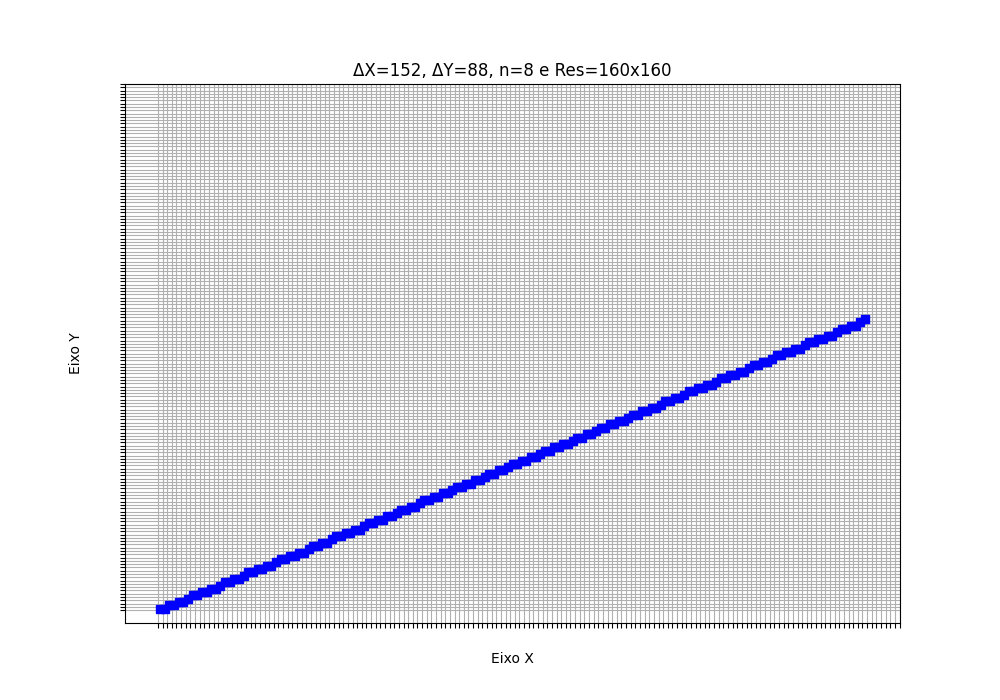


Figura 4. Fonte: Própria da dupla

Agora, iremos utilizar como exemplo a semirreta apresentada na aula do dia 10/09/2021,e nesse caso iremos utilizar a resolução da figura 2.

No exemplo da aula, tínhamos:

(X1, Y1) = (0,0)

(X2,Y2) = (9,3)

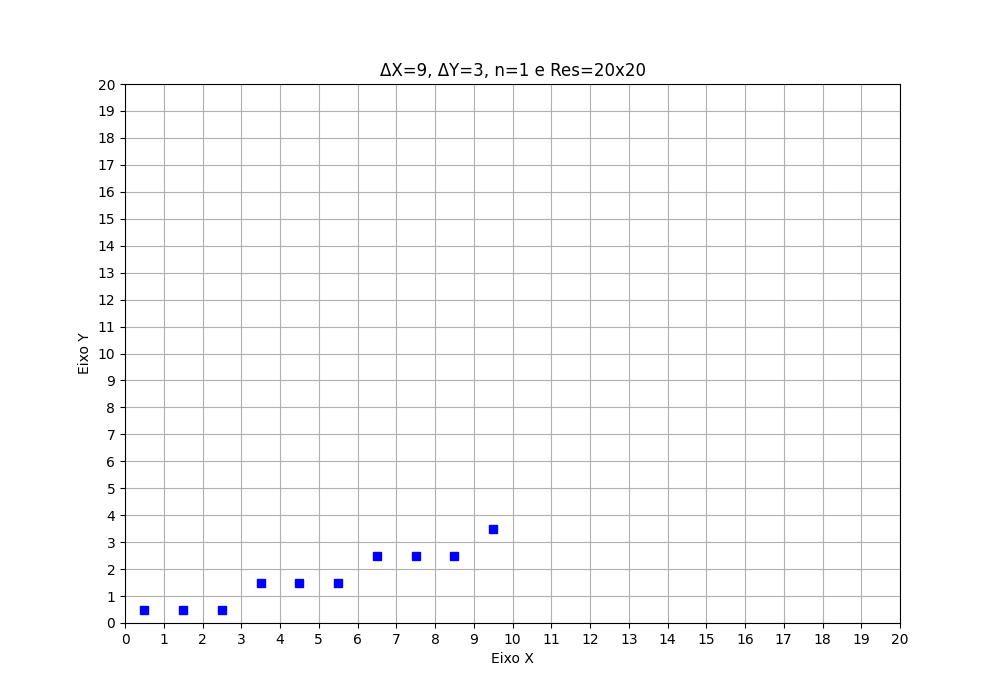


Figura 5. Fonte: Própria da dupla

3.2 Retas com **​ΔY>ΔX**

Para maiores variações no eixo Y, o algoritmo passou a variar em Y e calcular X a partir da equação X = (Y-B) / M. Com isso, foi possível obter as Figuras 6, 7 e 8 utilizando variações de 0 a 19 em y e de 0 a 11 em x.

Ou seja:

(X1,Y1)=(0,0)

(X1,Y1)=(11,19)

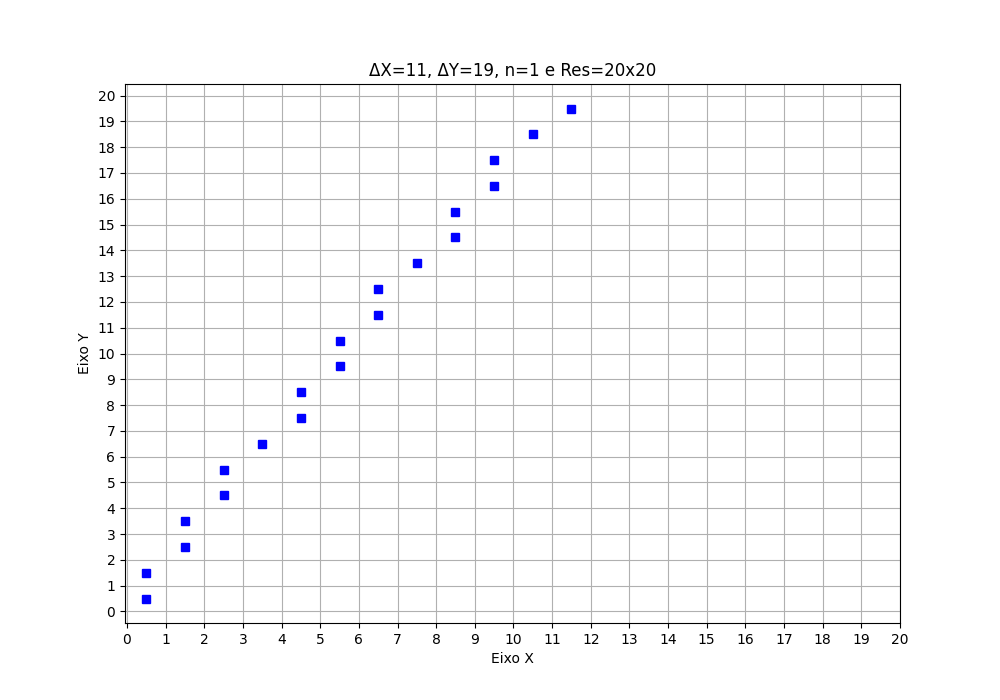


Figura 6. Fonte: Própria da dupla

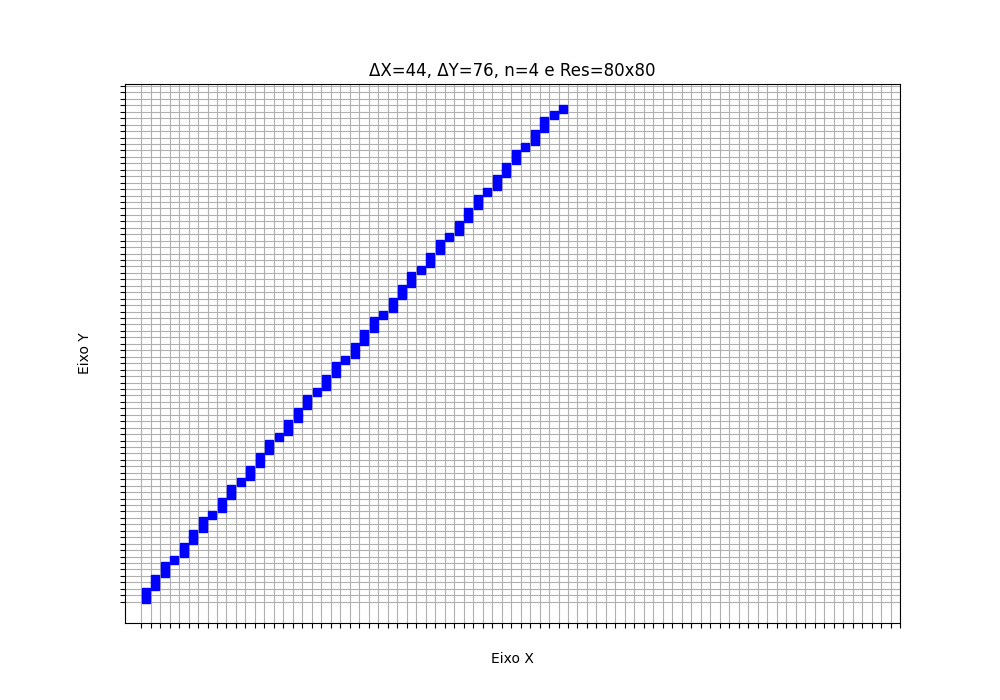
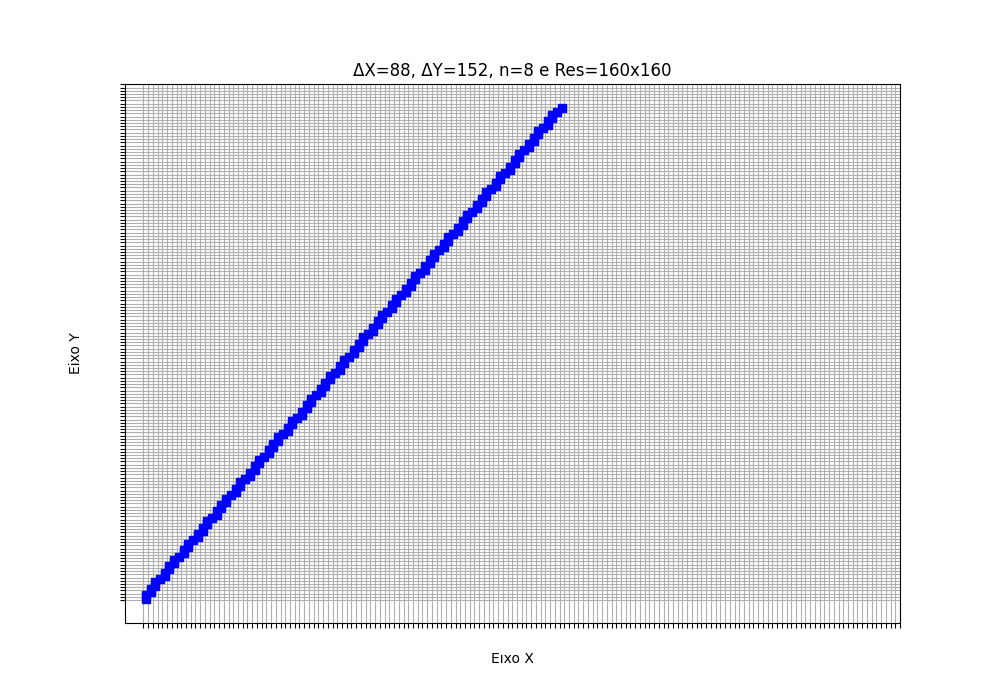


Figura 7. Fonte: Própria da dupla

Figura 8. Fonte: Própria da dupla

Agora, iremos utilizar como exemplo a semirreta apresentada na aula do dia 10/09/2021 com os valores de X2 e Y2 invertidos, tal que  **​ΔY>ΔX** ,e nesse caso também iremos utilizar a resolução da figura 2.

No exemplo da aula, tínhamos:

(X1, Y1) = (0,0)

(X2,Y2) = (3,9)

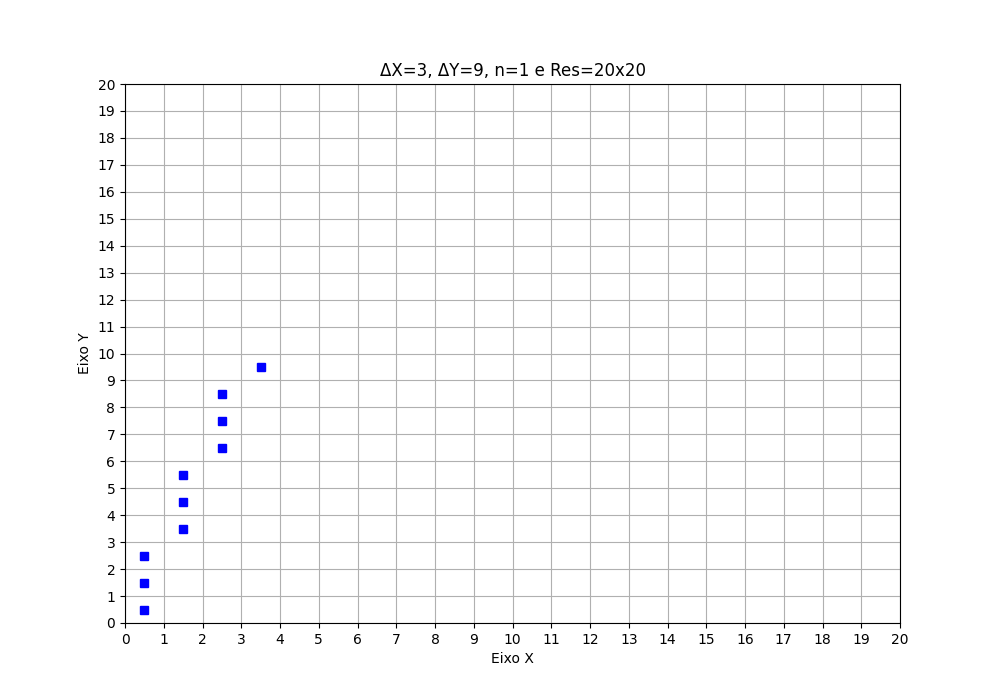


Figura 9. Fonte: Própria da dupla

3.3 Comparação das retas geradas

A partir da análise dos gráficos acima podemos concluir que o aumento da resolução faz com que as distorções das retas geradas diminuam, porém ainda é possível visualizar a borda dos pixels ao dar um zoom na imagem.

3.4 Retas ​totalmente Horizontais ou Verticais.

Para as retas horizontais, o algoritmo original funcionou normalmente, porém

com a adaptação com a variável N, foi necessário uma verificação adicional para definir se delta Y seria zero ou não. O mesmo foi feito no caso de delta X igual a zero. Gerando assim, a Figura 10 com 4 vezes mais resolução, Y(y1=20 e y2=20) não possuindo variação e X variando de 0 a 20 para uma resolução inicial de 20x20. A distorção na reta não é notada devido a organização dos pixels.

Já para retas horizontais, além da verificação por conta da variável N, foi necessário considerar M = 0 caso delta de X se igualasse a zero. Diante disso, como M=0 não era possível calcular X por X = (Y-B) / M, assim o ponto de X considerado foi o X final. Assim, foi possível gerar Figura 11 com N=4, considerando Y variando de 0 a 20 e X não possuindo variação em uma resolução inicial de 20x20.

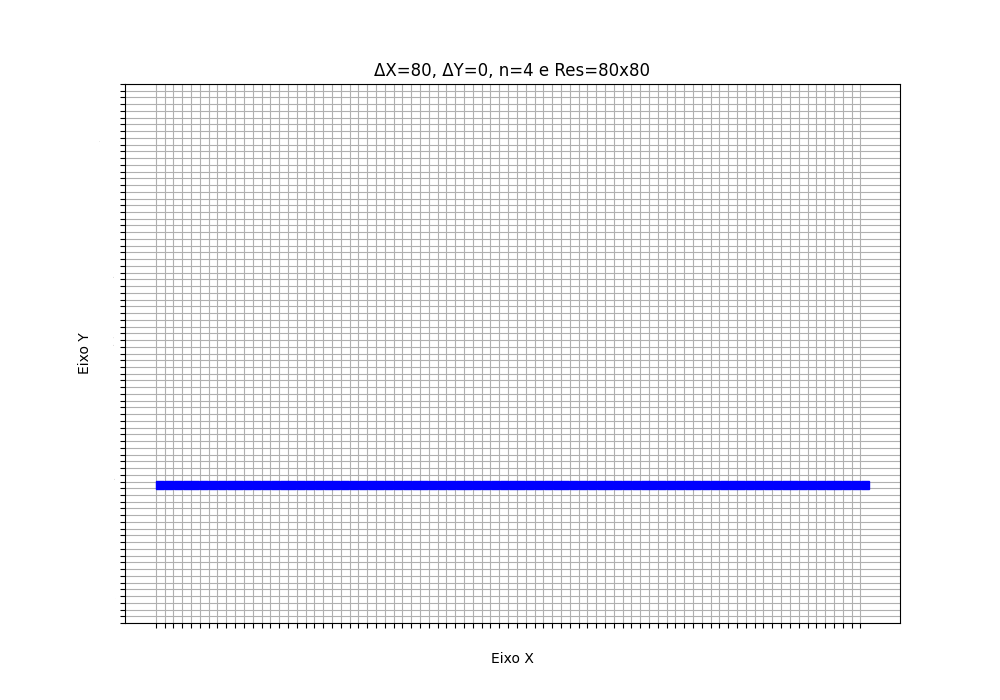


Figura 10. Fonte: Própria da dupla

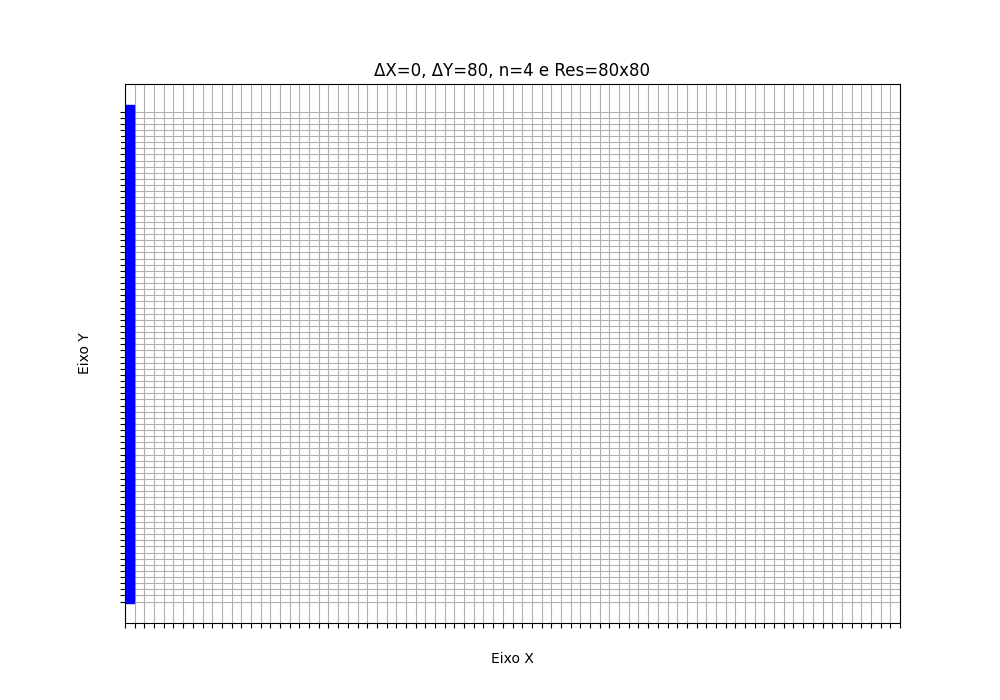


Figura 11. Fonte: Própria da dupla

**4. Introdução a rasterização de polígonos**

A rasterização de polígonos tem por objetivo atribuir cor aos pixels pertencentes ao polígono, e essa cor não é necessariamente uniforme, podendo variar de pixel para pixel.

****

Figura 12. Fonte: Professor João Manuel Brisson Lopes

**5.Objetivos.**

Implementar o algoritmo de rasterização de polígonos convexos que utiliza o algoritmo de rasterização de rentas.

Obter: Rasterização de 6 polígonos.

**6. Resultados da implementação do algoritmo de rasterização de polígonos.**

Analogamente ao algoritmo de rasterização de retas, o algoritmo rasterização de polígonos foi implementado em python e está disponível no seguinte link:

Rasterização de Triângulo 20x20:

<https://replit.com/@ArlindoAlcntara/Rasterizacao-de-Triangulo-20x20>

Rasterização de Triângulo 80X80

https://replit.com/@ArlindoAlcntara/Rasterizacao-de-Triangulo-80x80

Rasterização de Quadrado 20x20

<https://replit.com/@ArlindoAlcntara/Rasterizacao-de-Quadrado-20x20>

Rasterização de Quadrado 80x80

https://replit.com/@ArlindoAlcntara/Rasterizacao-de-Quadrado-80x80#main.py

Rasterização do hexágono 20x20

<https://replit.com/@ArlindoAlcntara/Rasterizacao-do-hexagono-20x20#main.py>

Rasterização do hexágono 80x80

https://replit.com/@ArlindoAlcntara/Rasterizacao-do-hexagono-80x80#main.py

​Para realizar a representação de pixels, utilizou-se das grades presentes em gráficos, assim cada quadrado no gráfico representa um pixel e o respectivo ponto marcado representa o centro do pixel. A nível de implementação foi considerado que todo pixel teria tamanho de 1 para qualquer resolução, assim, para aumentar a resolução de uma imagem, é necessário dobrar o tamanho de representação da reta em X e Y para o modelo adotado.

Para representar um aumento de resolução, foi utilizada uma constante N para representar quantas vezes a resolução seria aumentada, ou seja, mais pixels no mesmo espaço. Com isso, a Figura 13 é a reta original com resolução 1 e as figuras 14 representam resolução 4 vez maior que a Figura 13.

No triângulo equilátero, temos:

(X1,Y1)= (0,0)

(X2,Y2) = (7,14)

(X3,Y3) = (14,0)

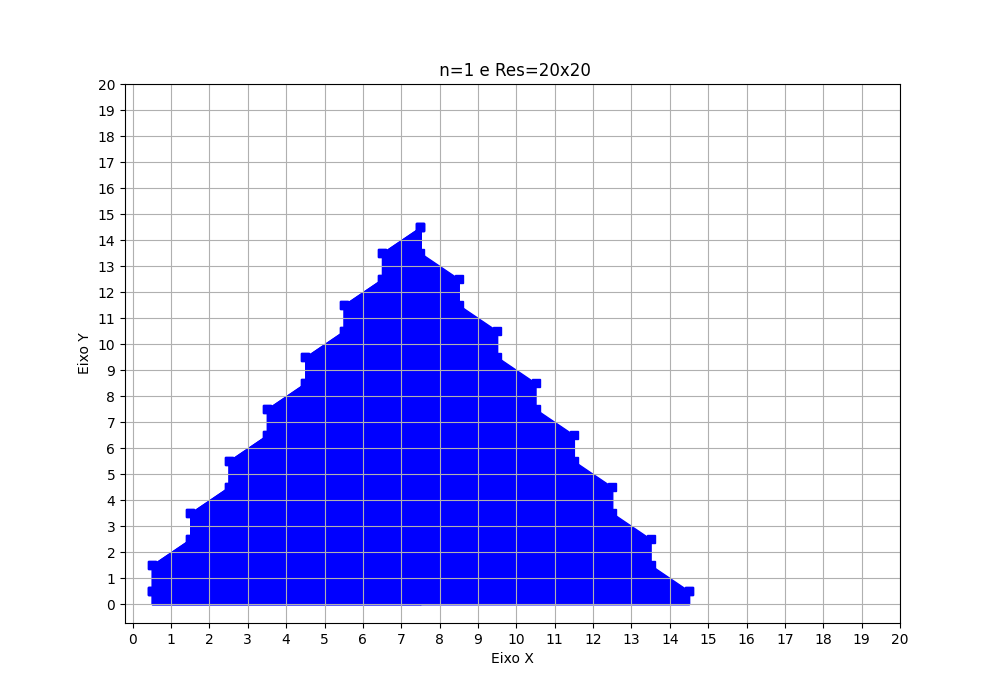
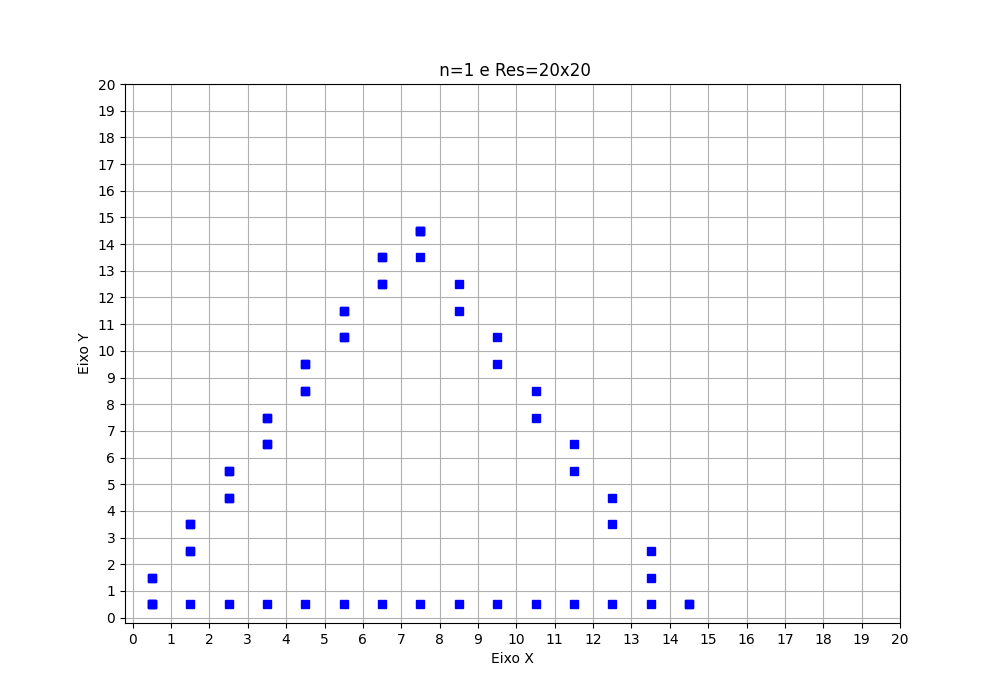
****

Figura 13. Fonte: Própria dos aluno

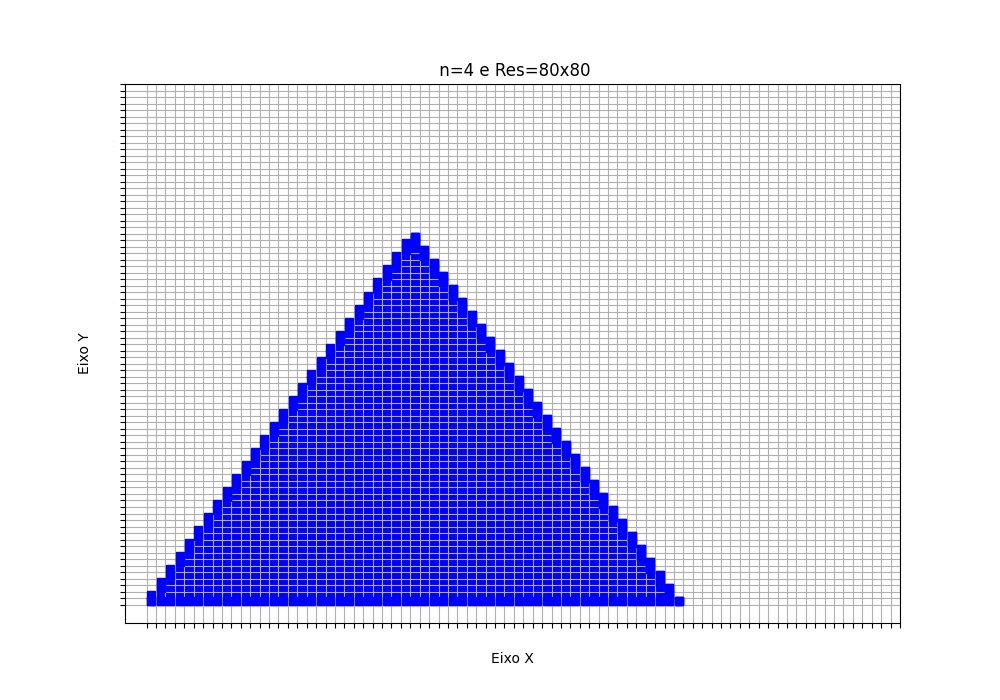
****

Figura 14. Fonte: Própria da dupla

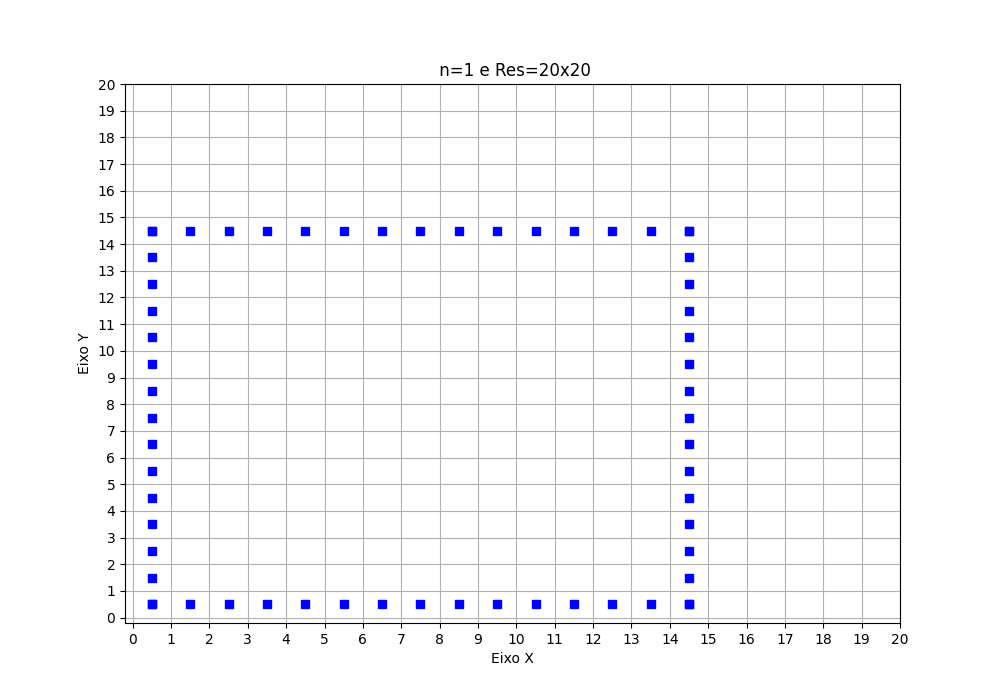
No quadrado, temos:

(X1,Y1)= (0,0)

(X2,Y2) = (0,14)

(X3,Y3) = (14,14)

(X4,Y4) = (14,0)



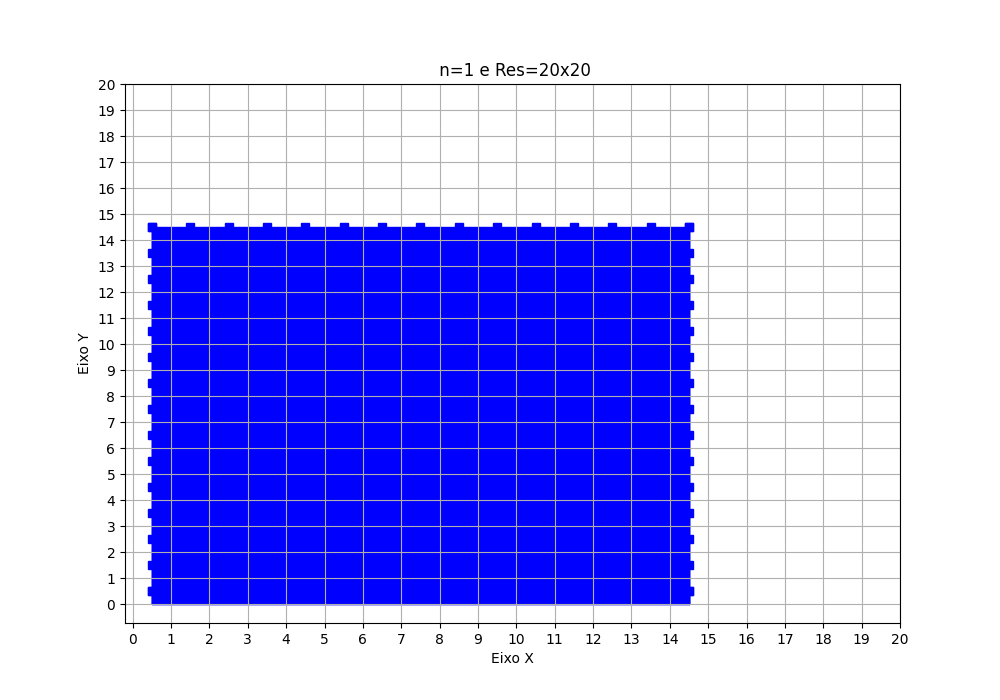
****

Figura 15. Fonte: Própria da dupla

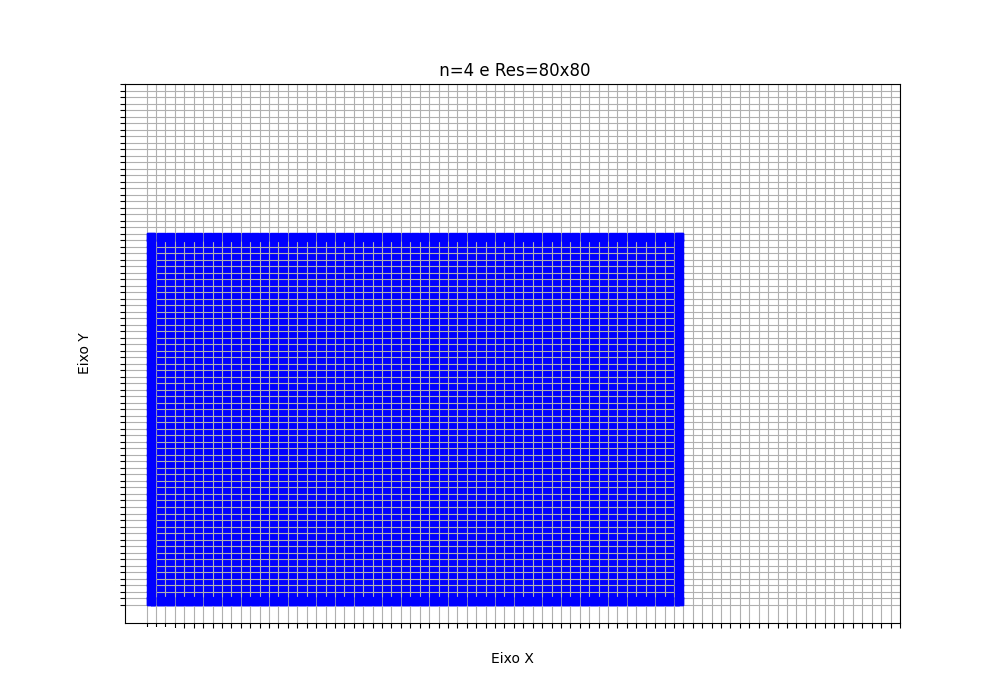
****

Figura 16. Fonte: Própria da dupla

No hexágono, temos:

(X1,Y1) = (4,3)

(X2,Y2) = (2,7)

(X3,Y3) = (4,12)

(X4,Y4) = (10,12)

(X5,Y5) = (12,7)

(X6,Y6) = (10,3)

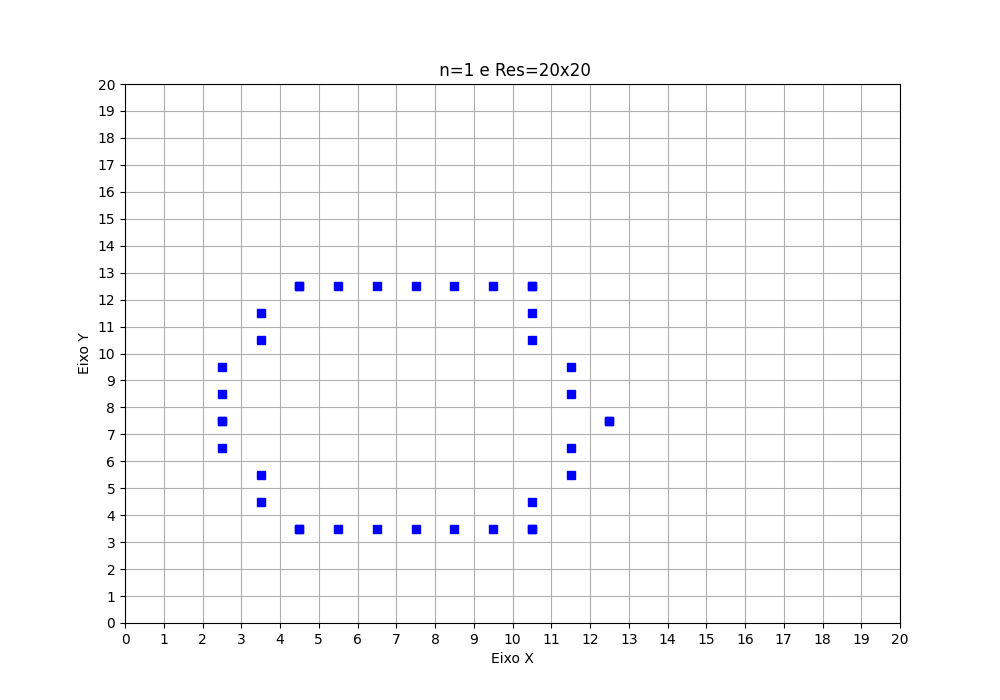


Figura 17. Fonte: Própria da dupla

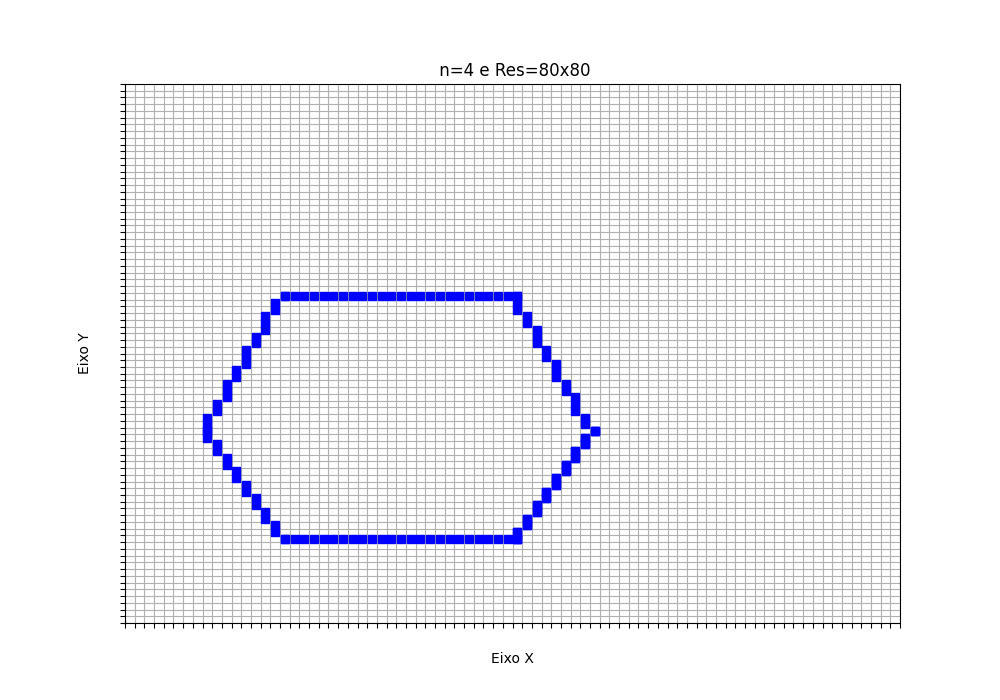


Figura 18. Fonte: Própria da dupla

**7. Conclusão**

Ao decorrer da execução do trabalho foi preciso compreender bem os processos algébricos e geométricos abordados pela atividade, e realizar a atividade com o suporte do conhecimento de programação na linguagem Python e suas bibliotecas, adquirido previamente. Nessa perspectiva, o processo mostrou à dupla que mais do que saber programar (que é essencial para a área da cadeira), ter noção “espacial” e “matemática” de estruturas como “matrizes” é importante para saber como se representar determinada situação que exige olhares de diferentes perspectivas.