Desenhando objetos

PyGame

Área de desenho

Antes de iniciarmos os exemplos desta aula, temos que explicar alguns conceitos.

Surface são superficies para desenho. Surfaces podem ser 2D ou 3D, podem ser armazenadas em memória do sistema ou da placa de vídeo e também ter várias profundidades de cores.

O PyGame possui a classe ***pygame.Surface*** utilizada para representar imagens e a classe ***pygame.Rect*** para armazenar coordenadas retangulares.

Pense no Surface como uma folha da papel em branco na qual você pode desenhar o que quiser. Eles podem conter imagens também.

Os Rects são uma representação de uma área retangular que o seu Surface engloba.

PyGame

Desenhando objetos

O PyGame possui o módulo ***pygame.draw*** para desenhar formas. Veja alguns métodos.

Método	Descrição
pygame.draw.rect	Desenha uma forma de retângulo.
pygame.draw.polygon	Desenha uma forma com qualquer número de lados.
pygame.draw.circle	Desenha um círculo em torno de um ponto.
pygame.draw.ellipse	Desenha uma forma redonda dentro de um retângulo.
pygame.draw.arc	Desenha uma seção parcial de uma elipse.
pygame.draw.line	Desenha um segmento de reta.
pygame.draw.lines	Desenha vários segmentos de retas.
pygame.draw.aaline	Desenha linhas finas Anti-Aliasing.
pygame.draw.aalines	Desenha uma sequência conectada de linhas Anti-Aliasing.

O anti-aliasing é um método de redução de serrilhamento, que é o efeito em forma de serra que se cria ao desenhar uma reta inclinada em um computador. Uma vez que a divisão mínima num monitor é de píxeis, surge o aparecimento dos "dentes" da serra ao longo da reta desenhada.

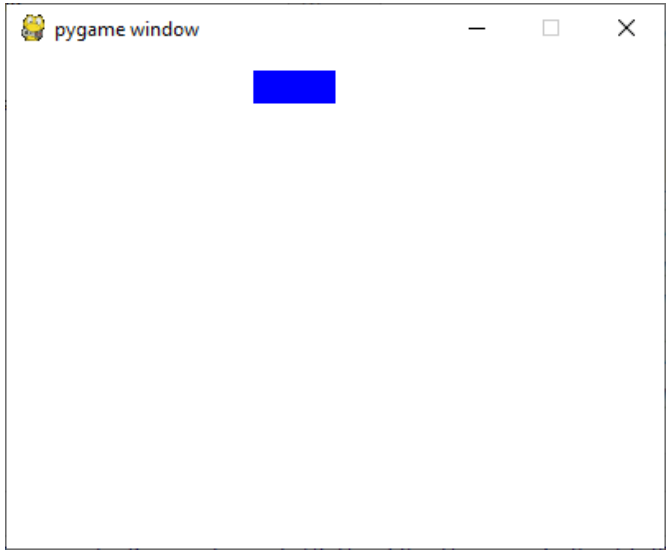
PyGame

Desenhando objetos

Para desenhar um retângulo, adicione a linha a seguir, entre os métodos *screen.fill* e *display.flip* do nosso exemplo da aula anterior.

Vamos desenhar um retângulo que inicia em 150 no eixo x, 10 no eixo y e tenha a lagura de 50 pixels e altura de 20 pixels.

```
# Preenchendo a superfície com branco
screen.fill((255, 255, 255))
# Desenha um retângulo azul na tela
# (0, 0, 255): Cor azul
# [x, y, largura, altura]
pygame.draw.rect(screen, (0, 0, 255), [150, 10, 50, 20])
pygame.display.flip()
```

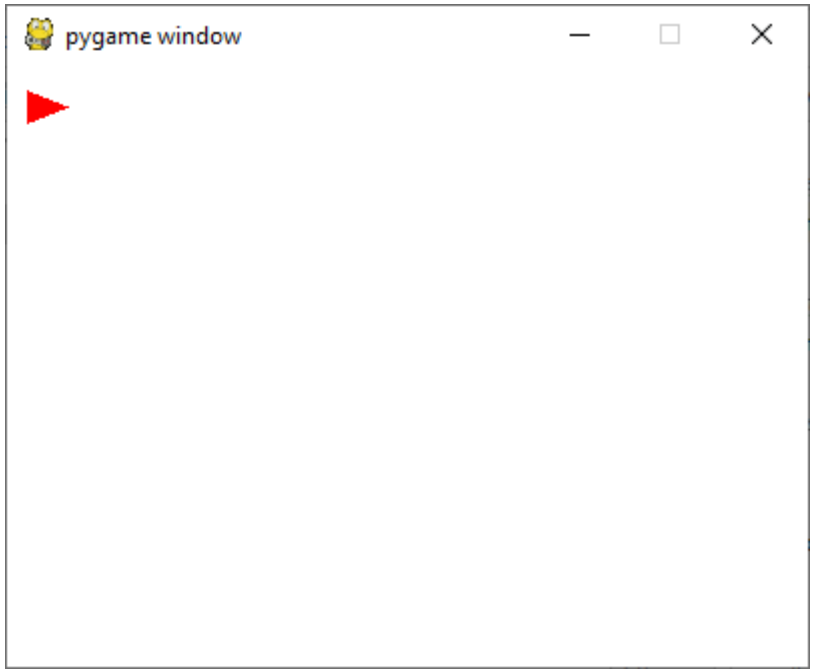


PyGame

Desenhando objetos

Agora vamos desenhar um polígono. O polígono necessita de ao menos 3 pontos de coordenadas x, y.

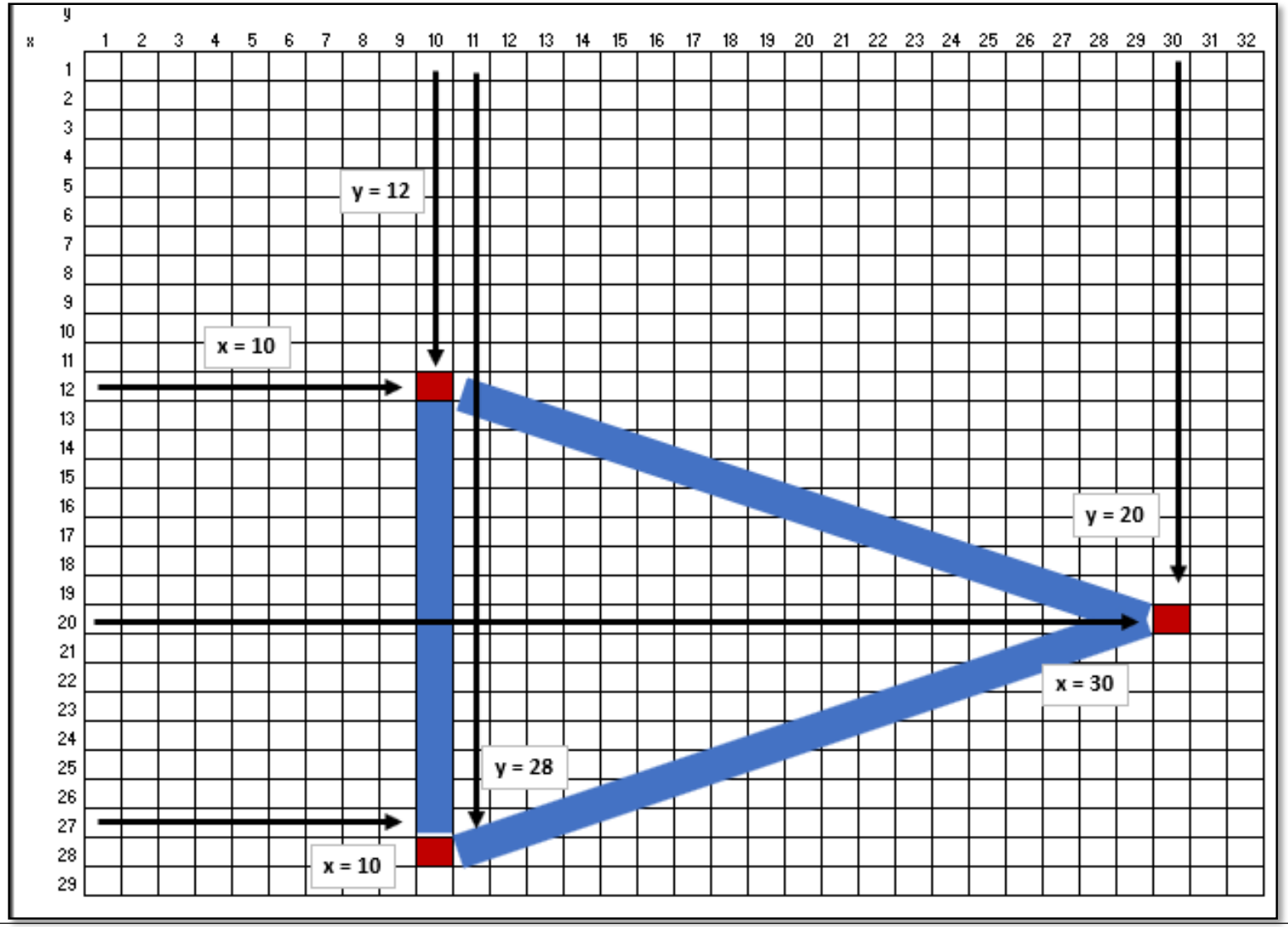
```
screen.fill((255, 255, 255))
# Desenha um polígono vermelho na tela
# (255, 0, 0): Cor vermelha
# [(x,y), (x,y), (x,y]: Lista contendo as tuplas com as coordenadas
pygame.draw.polygon(screen, (255, 0, 0), [(10,12), (10,28), (30,20)])
# Atualiza o conteúdo de toda a tela
pygame.display.flip()
```



PyGame

Desenhando objetos

Veja como foi desenhado o polígono. A base para o polígono foram as coordenadas (x=10, y=12) (x=10, y=28) (x=30, y=20) Representados pelos pontos vermelhos.



PyGame

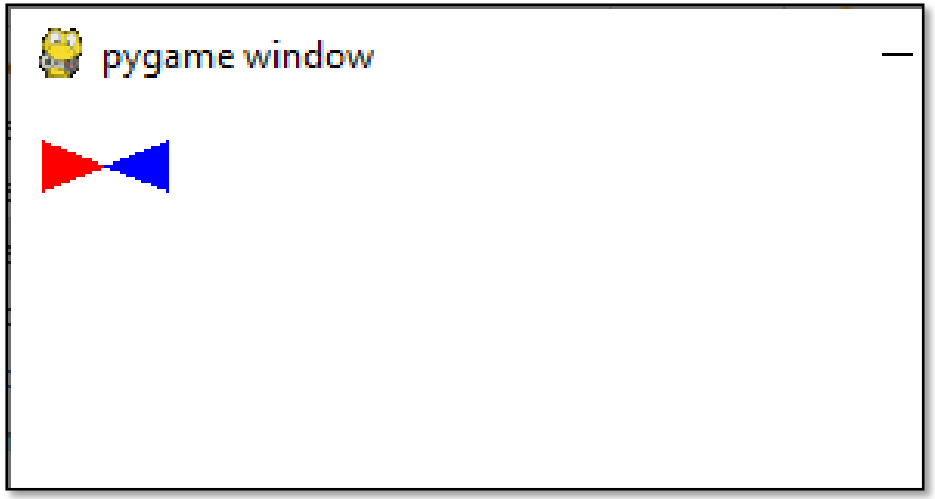
Desenhando objetos

Agora adicione mais um polígono ao nosso exemplo, usando agora, a cor azul conforme imagem ao lado.

Para isso repita a linha do polígono, conforme exibido abaixo e defina os valores que estão preenchidos com interrogação.

Pause o video e tente fazer.

Caso não consiga, dê uma olhada no arquivo exemplo03_poligono.py.




```
pygame.draw.polygon(screen, (255, 0, 0), [(10,12), (10,28), (30,20)])
pygame.draw.polygon(screen, (?, ?, ?), [(?, ?), (?, ?), (?, ?)])
```

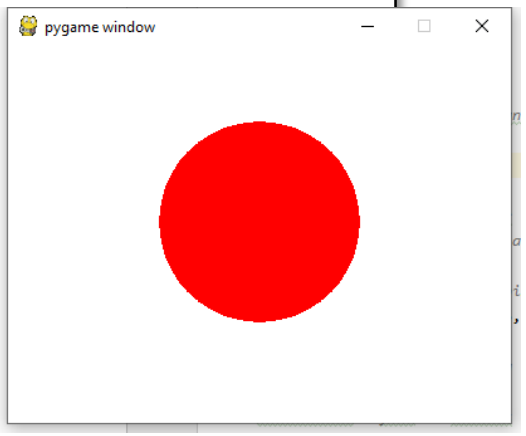
PyGame

Desenhando objetos

Agora vamos desenhar um círculo. Para isso, Podemos usar pygame.draw.circle informando:

pygame.draw.circle(superfície, (cor), (x,y), raio)

```
# Desenha um círculo Vermelho
# Devemos informar a superfície onde vamos desenhar, a cor, uma tupla com x e y para o Centro e o Raio
# (255, 0, 0) é uma tupla para a cor vermelho
# (200, 140), onde X é 200 e y, 140, lembrando que nossa tela tem x=400 e y=300
# Vamos definir um raio de 80 pixels
pygame.draw.circle(screen, (255, 0, 0), (200,140), 80)
```



PyGame

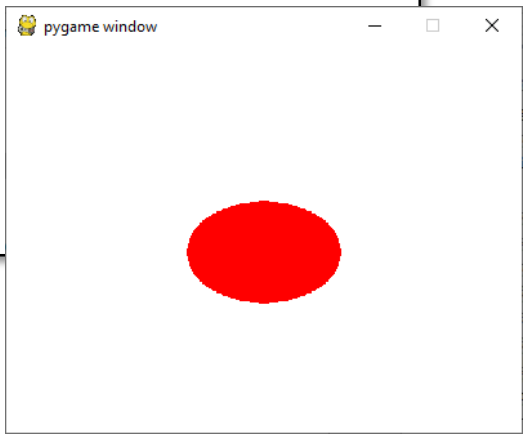
Desenhando objetos

Agora vamos desenhar uma elipse. Para isso, Podemos usar pygame.draw.ellipse informando:

pygame.draw.ellipse(superfície, (cor), retângulo)

O retângulo é um objeto Rect, que armazena coordenadas retangulares. Ele indica a posição e as dimensões da elipse, a elipse será centralizada dentro do retângulo e delimitada por ele.

```
# Desenha uma elipse Vermelha
# Devemos informar a superfície onde vamos desenhar, uma tupla com a cor, e um objeto Rect, que armazena
# coordenadas retangulares e indica a posição e as dimensões da elipse, a elipse será centralizada dentro
# do retângulo e delimitada por ele.
# (255, 0, 0) é uma tupla para a cor vermelho
# Os parâmetros de um Rect são:
# Rect(esquerda, topo, largura, altura)
retangulo = pygame.Rect(140, 120, 120, 80)
pygame.draw.ellipse(screen, (255, 0, 0), retangulo)
```



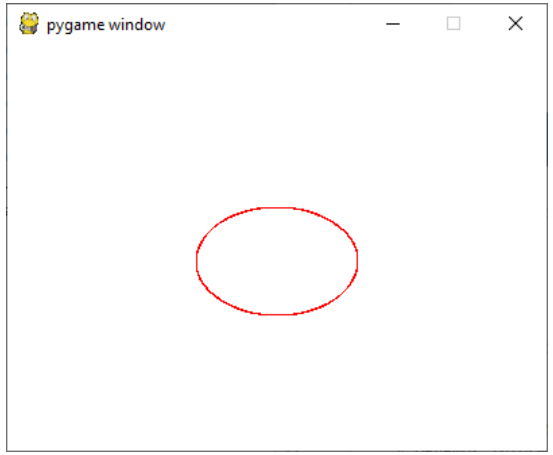
PyGame

Desenhando objetos

Agora vamos desenhar um arco. Para isso, Podemos usar pygame.draw.arc informando:

pygame.draw.arc(superfície, (cor), retângulo, início, fim)

```
# Desenha um arco Vermelho
# Devemos informar a superfície onde vamos desenhar, uma tupla com a cor, um objeto Rect e
# um valor inicial e final para o ângulo do arco em radianos
retangulo = pygame.Rect(140, 120, 120, 80)
pygame.draw.arc(screen, (255, 0, 0), retangulo, 10, 50)
```



PyGame

Desenhando objetos

Agora vamos desenhar uma linha. Para isso, Podemos usar pygame.draw.line informando:

pygame.draw.line(superfície, (cor), (posição inicial), (posição final))

Posição inicial e posição final são duas tuplas com valores para x e y.

```
# A linha vai começar na posição x=140 e y=140 e vai finalizar na posição x=200 e y=240
pygame.draw.line(screen, (255, 0, 0), (140, 140), (200, 240))
```

PyGame

Desenhando objetos

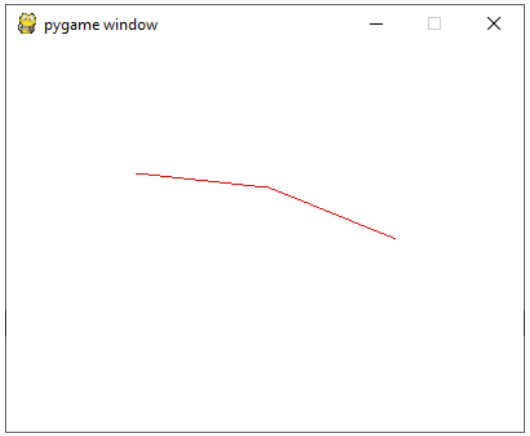
Vamos agora desenhar vários segmentos de linha reta. Para isso podemos usar pygame.draw.lines.

pygame.draw.lines(superfície, (cor), Fechado, [(x1, y1), (x2, y2)...])

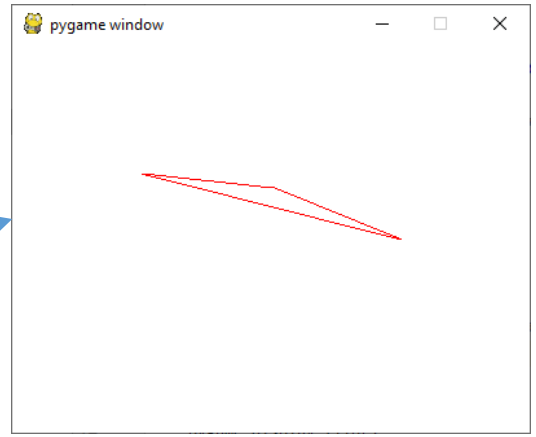
O ultimo parâmetro é uma lista com pelo menos duas tuplas, definindo as coordenadas para que a linha seja traçada.

O "Fechado" recebe True ou False. Se for informado True, haverá uma linha após a ultima coordenada conectando à primeira coordenada, caso contrário, não será feita esta linha "fechando".

```
# A linha vai começar na posição x=140 e y=140 e vai finalizar na posição x=200 e y=240
pygame.draw.lines(screen, (255, 0, 0), False, [(100, 100), (200, 110), (300, 150)])
```



Mudando de False para True.



PyGame

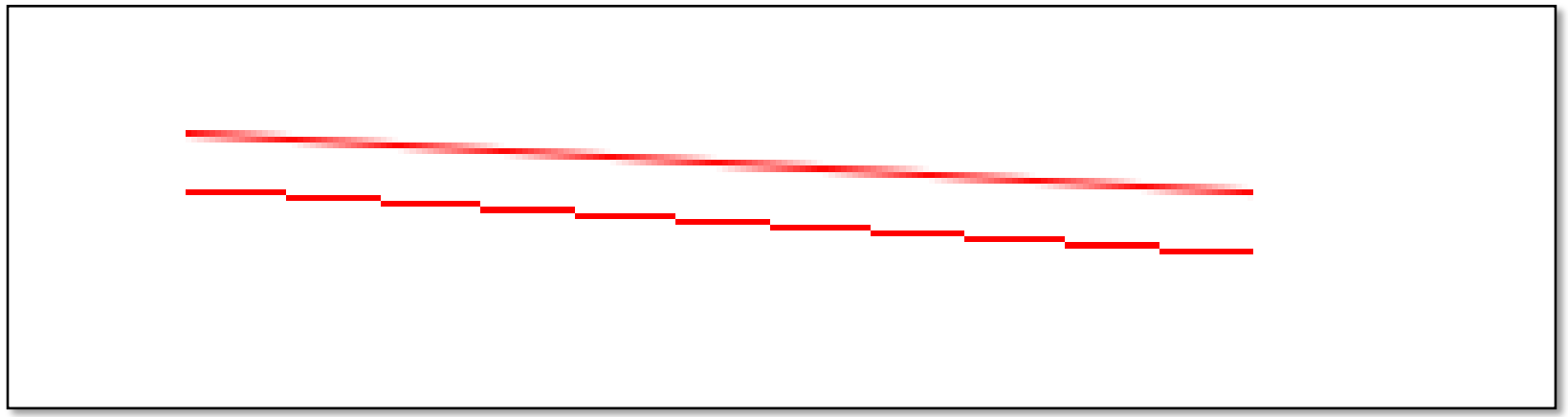
Desenhando objetos

Agora vamos desenhar uma linha reta antialias. Para isso, Podemos usar pygame.draw.aaline informando:

pygame.draw.aaline(superfície, (cor), (posição inicial), (posição final))

Vamos desenhar duas linhas, uma usando “lines” e outra “aaline” para você ver a diferença.

```
pygame.draw.aaline(screen, (255, 0, 0), (100, 100), (280, 110))
pygame.draw.line(screen, (255, 0, 0), (100, 110), (280, 120))
```

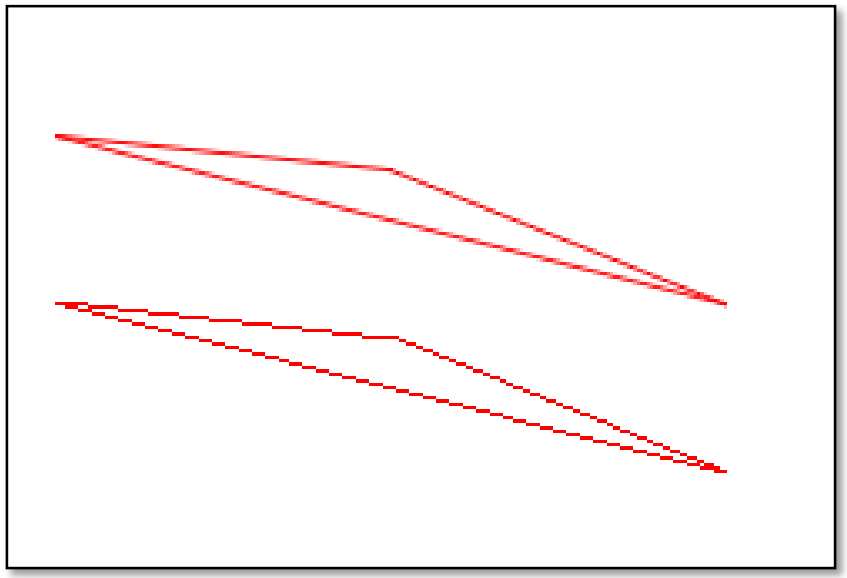


PyGame

Desenhando objetos

O aalines funciona como o lines. Com ele podemos criar linhas antialias. Os parâmetros são similares ao método lines. Veja um exemplo usando lines e aalines.

```
pygame.draw.aalines(screen, (255, 0, 0), True, [(100, 100), (200, 110), (300, 150)])
pygame.draw.lines(screen, (255, 0, 0), True, [(100, 150), (200, 160), (300, 200)])
```



FIM