```
        # construit une chaîne pour le titre de la fenêtre et le journal
        coordText = '('+str(nSX)+','+str(nSY)+')-
                    ('+str(nEX)+','+str(nEY)+')'
        # rendu de l'image ligne par ligne (dure plus longtemps) ?
        if args.renderlines == 'y':
            pygame.display.update();
            pygame.display.set_caption(coordText)

        # envoie les coordonnées dans le journal
        fnAppend2Log(coordText)

    # retourne les coordonnées finales calculées
    return (nSX, nSY, nEX, nEY);

# définit les arguments de la ligne de commande :
parser = argparse.ArgumentParser(description='Render shape')
parser.add_argument('-s', action='store', dest='scale', type=int,
                    default=2, help='Render size, default=2, 200x200px)')
parser.add_argument('-t', action='store', dest='step', type=int,
                    default=5,
                    help='Diminuez l'écart pour des lignes plus denses')
parser.add_argument('-r', action='store', dest='renderlines',
                   choices=('o','n'), default='o',
                   help='Afficher par ligne (O) ou l'objet terminé (n)')
args = parser.parse_args()

# Définit les variables qui seront nécessaires
sz = 100*args.scale # taille en pixels horiz x vert d'un quart de l'image
iterations = sz +5 # nombre de lignes à calculer par quartier
lineColour = 0,0,255 # la couleur de la ligne à dessiner (bleu)

# ouvre un écran pygame sur lequel sera fait le rendu de nos objets
# la taille de l'image double de l'objet à générer car nous avons 4 quarts
screen = pygame.display.set_mode([sz*2,sz*2],0,32)

# Dessin des lignes, quartier, renvoyant les paires de coordonnées
# Les coordonnées de début sont celles de fin du dernier quart généré
sx, sy, ex, ey = fnPlotLines('Top left', sz, 0, sz, sz, 0, 1, -1, 0 )
sx, sy, ex, ey = fnPlotLines('Bottom left', ex, ey, sx, sy, 1, 0, 0, 1 )
sx, sy, ex, ey = fnPlotLines('Bottom right', ex, ey, sx, sy, 0, -1, 1, 0 )
sx, sy, ex, ey = fnPlotLines('Top right', ex, ey, sx, sy, -1, 0, 0, -1 )

# si le rendu de chaque ligne est supprimé, affichage de l'image finale
if args.renderlines == 'n':
    pygame.display.update();

# enregistre l'image générée dans un fichier
pygame.image.save(screen, 'lineimage.png')

# affiche le résultat pendant 10 secondes
pygame.time.wait(10000)
```

Essayez d'ajouter un argument supplémentaire pour qu'il soit possible de désactiver l'écriture dans le fichier journal. Cela va améliorer la durée du rendu. Indice : tout comme pour les arguments, vous devrez ajouter deux instructions If car l'écriture dans le journal est faite à deux endroits différents dans le code.

Quelques autres idées d'arguments de ligne de commande :

- Laisser l'utilisateur choisir le nom du fichier pour l'image de sortie
- Spécifier les couleurs de lignes et d'arrière-plan à partir d'une liste prédéfinie (noir, blanc, rouge, vert, bleu)
- Entrer en mode démo qui boucle sur le code, produisant des surfaces avec des couleurs aléatoires