GERER LES ÉVÉNEMENTS

Tous les événements qui se présentent sont stockés au fur et à mesure dans une file d'attente ; ils seront traités dans l'ordre. On peut éventuellement les filtrer, si le besoin se manifeste.

C'est dans ces 3 lignes que sont traités les événements :

```
for event in pygame.event.get() :
    if event.type == pygame.QUIT :
        launched= False
```

L'événement à gérer est récupéré dans : pygame.event.get() et placé dans la variable event
La boucle while dans laquelle se situent ces 3 lignes va récupérer tous les événements qui se présentent à la suite.
event possède l'attribut type qui permet de savoir à quel type d'événement on a affaire.
Les différents types d'événements qui peuvent se présenter sont les suivants :

QUIT	none
ACTIVEEVENT	gain, state
KEYDOWN	unicode, key, mod
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value
JOYBALLMOTION	joy, ball, rel
JOYHATMOTION	joy, hat, value
JOYBUTTONUP	joy, button
JOYBUTTONDOWN	joy, button
VIDEORESIZE	size, w, h
VIDEOEXPOSE	none
USEREVENT	code

Par exemple l'événement QUIT correspond à l'action sur la croix en haut à droite qui entraîne la fermeture de la fenêtre
On peut même créer ses propres événements (USEREVENT)

1 – Gestion de l'événement VIDEORESIZE

Cet événement correspond à une action de redimensionnement de la fenêtre

Pour que la fenêtre soit redimensionnable, il faut que la commande
screen = pygame.display.set_mode(resolution)
comporte un deuxième argument : pygame.RESIZABLE
Souhaitons qu'à tout moment la fenêtre affiche la valeur de la résolution, initialement 640x480 :
Pour ce faire il faut commencer par charger une police de caractères .
Sur Windows (mais pas sur Linux) on peut par exemple charger la police de caractères Arial :

```
arial_font = pygame.font.SysFont(« Arial », 30)   (taille de 30 px)
dimensions_text = arial_font.render(« {} ».format(resolution, True, blank)
screen.blit(dimensions_text,[10,10])
```

La valeur de la résolution initialement 640x480, sera donc affichée en coordonnées 10, 10

Ecrire : from pygame.locals import * au lieu de import pygame éviterait d'écrire pygame partout (devant RESIZABLE, QUIT ou VIDEORESIZE)

Il faut réécrire dans la boucle while le bloc :

```
elif event.type == pygame.VIDEORESIZE :
    screen.fill(bleu)
    dimensions_text = arial_font.render("{}x{}".format(event.w,event.h), True, blank)
    screen.blit(dimensions_text, [10,10])
    pygame.display.flip()
```

afin que l'événement généré soit traité lorsqu'on redimensionne la fenêtre : c'est lorsque la souris sera relâchée que la nouvelle valeur de la résolution apparaîtra

On note que dans le format apparaissent les paramètres event.w et event.h qui donnent les nouvelles valeurs de la résolution

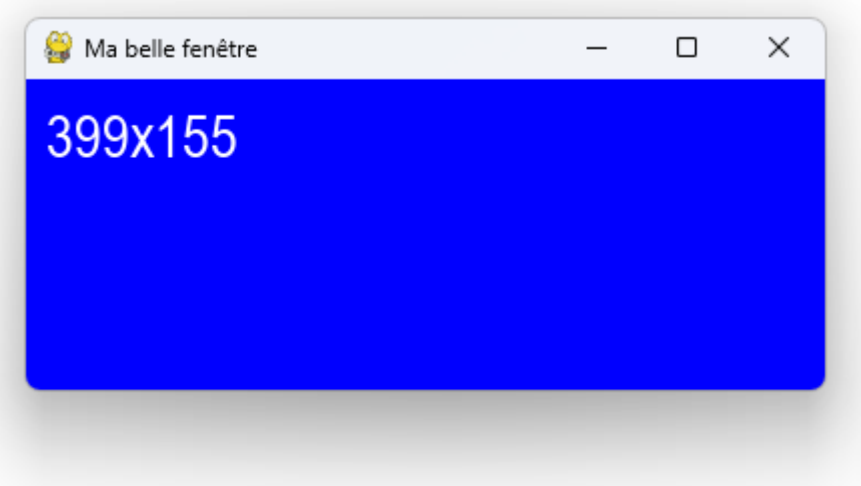
```
#coding:utf-8
import pygame
#from pygame.locals import *                                    #alternative


pygame.init()                                                  #constructeur
resolution = (640,480)
bleu    = (0,0,255)                                        # creation des couleurs
blank = (255,255,255)
black  = (0,0,0)
pygame.display.set_caption("Ma belle fenêtre")
screen  =  pygame.display.set_mode(resolution,pygame.RESIZABLE)
screen.fill(bleu)                                      #  pour colorier la fenêtre en bleu


arial_font= pygame.font.SysFont("arial", 30)
dimensions_text = arial_font.render("{}".format(resolution), True, blank)
screen.blit(dimensions_text, [10,10])
pygame.display.flip()

launched = True
while launched :                              #  boucle qui permet de laisser la surface
  for event in pygame.event.get() :     # gestion des événements
    if event.type == pygame.QUIT :
      launched= False
    elif event.type == pygame.VIDEORESIZE :
      screen.fill(bleu)
      dimensions_text = arial_font.render("{}x{}".format(event.w,event.h), True, blank)
      screen.blit(dimensions_text, [10,10])
      pygame.display.flip()
```

Fenêtre initiale : indique sa résolution



Fenêtre redimensionnée : indique sa nouvelle résolution

2 – Gestion de l'événement CLAVIER (appui touche)

QUIT	none
ACTIVEEVENT	gain, state
KEYDOWN	unicode, key, mod
KEYUP	key, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, button
MOUSEBUTTONDOWN	pos, button
JOYAXISMOTION	joy, axis, value
JOYBALLMOTION	joy, ball, rel
JOYHATMOTION	joy, hat, value
JOYBUTTONUP	joy, button
JOYBUTTONDOWN	joy, button
VIDEORESIZE	size, w, h
VIDEOEXPOSE	none
USEREVENT	code

Il existe des constantes liées aux différentes touches du clavier, que l'on peut retrouver dans un fichier fourni par le système :

KeyASCII	ASCII	Common Name
K_BACKSPACE	\b	backspace
K_TAB	\t	tab
K_CLEAR		clear
K_RETURN	\r	return
K_PAUSE		pause
K_ESCAPE	^[	escape
K_SPACE		space
K_EXCLAIM	!	exclaim
K_QUOTEDBL	"	quotedbl
K_HASH	#	hash
K_DOLLAR	$	dollar
K_AMPERSAND	&	ampersand
K_QUOTE		quote
K_LEFTPAREN	(	left parenthesis
K_RIGHTPAREN	)	right parenthesis
K_ASTERISK	*	asterisk
K_PLUS	+	plus sign
K_COMMA	,	comma
K_MINUS	-	minus sign
K_PERIOD	.	period
K_SLASH	/	forward slash
K_0	0	0
K_1	1	1
K_2	2	2
K_3	3	3
K_4	4	4

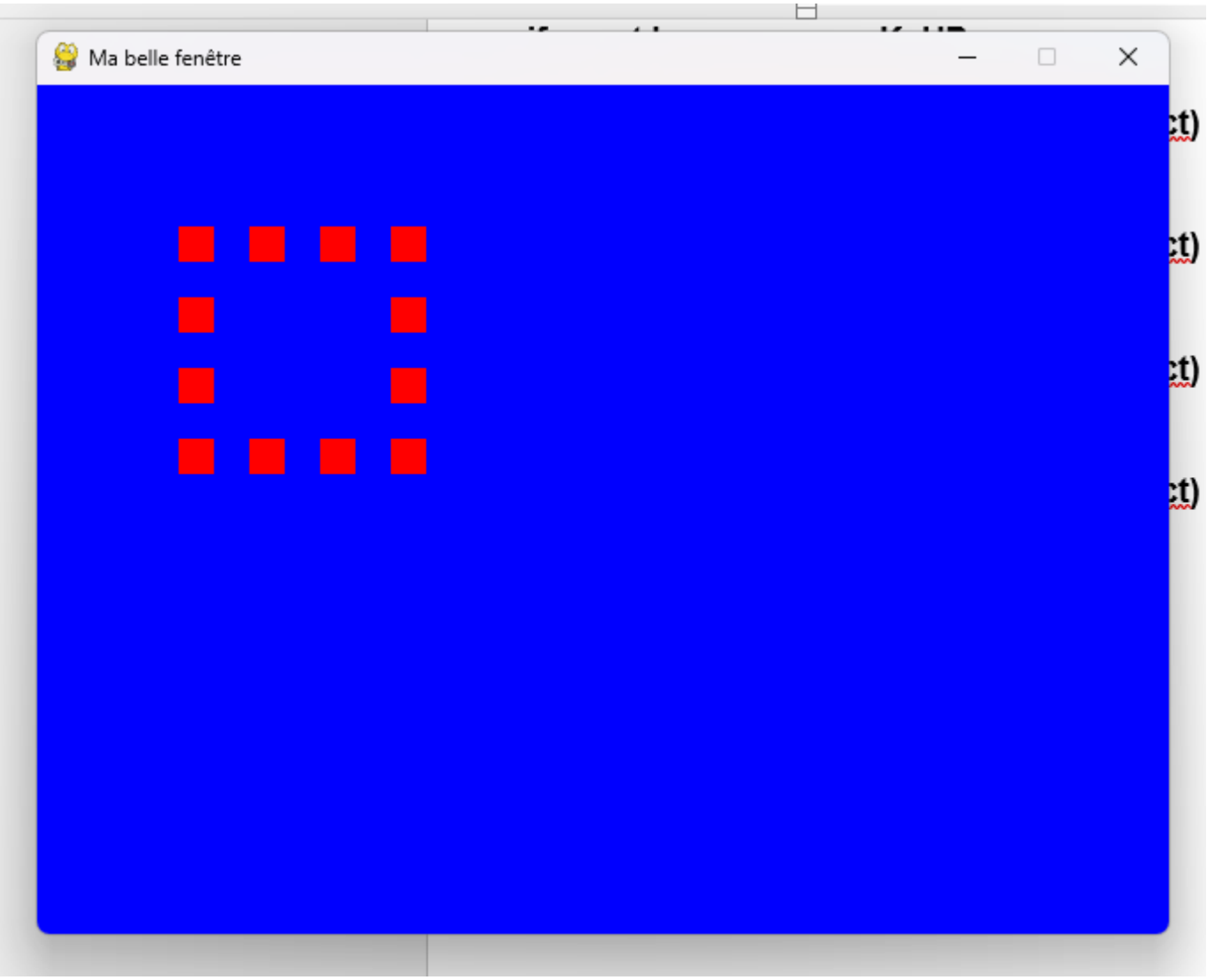
etc

On peut ainsi déplacer un petit carré rouge en pressant alternativement les touches K_DOWN, K_RIGHT, K_UP et K_LEFT 3 fois chacune :

```
import pygame
#from pygame.locals import *                    #alternative
bleu=(0,0,255)
rouge=(255,0,0)
pygame.init()                                  #constructeur
resolution = (640,480)
pygame.display.set_caption("Ma belle fenêtre")
screen  =  pygame.display.set_mode(resolution)

screen.fill(bleu)                              #  pour colorier la fenêtre en bleu
myrect = pygame.Rect(80,80,20,20)     #constructeur de l'objet myrect
pygame.draw.rect(screen, rouge, myrect)     # rafraichissement
pygame.display.flip()

launched = True
while launched :                               #  boucle qui permet de laisser la surface
  for event in pygame.event.get() :                        # gestion des événements
    if event.type == pygame.QUIT :
      launched= False
    elif event.type == pygame.KEYDOWN :
      if event.key==pygame.K_UP :
        myrect.y -= 40
      elif event.key==pygame.K_DOWN :
        myrect.y += 40
      elif event.key==pygame.K_LEFT :
        myrect.x -= 40
      elif event.key==pygame.K_RIGHT :
        myrect.x += 40
      else :
        print("Autre touche")
    # screen.fill(bleu)
    pygame.draw.rect(screen, rouge, myrect)
    pygame.display.flip()
```



Pour que les traces précédentes ne soient pas visibles et qu'on ne voie le carré qu'une fois à sa nouvelle position , il suffit de décommenter la ligne
screen.fill(bleu)
en avant-avant-dernière position

si l'on frappe une touche différente des 4 spécifiées, le texte « Autre touche » apparaîtra dans le terminal.

3 – Gestion de l'événement Souris

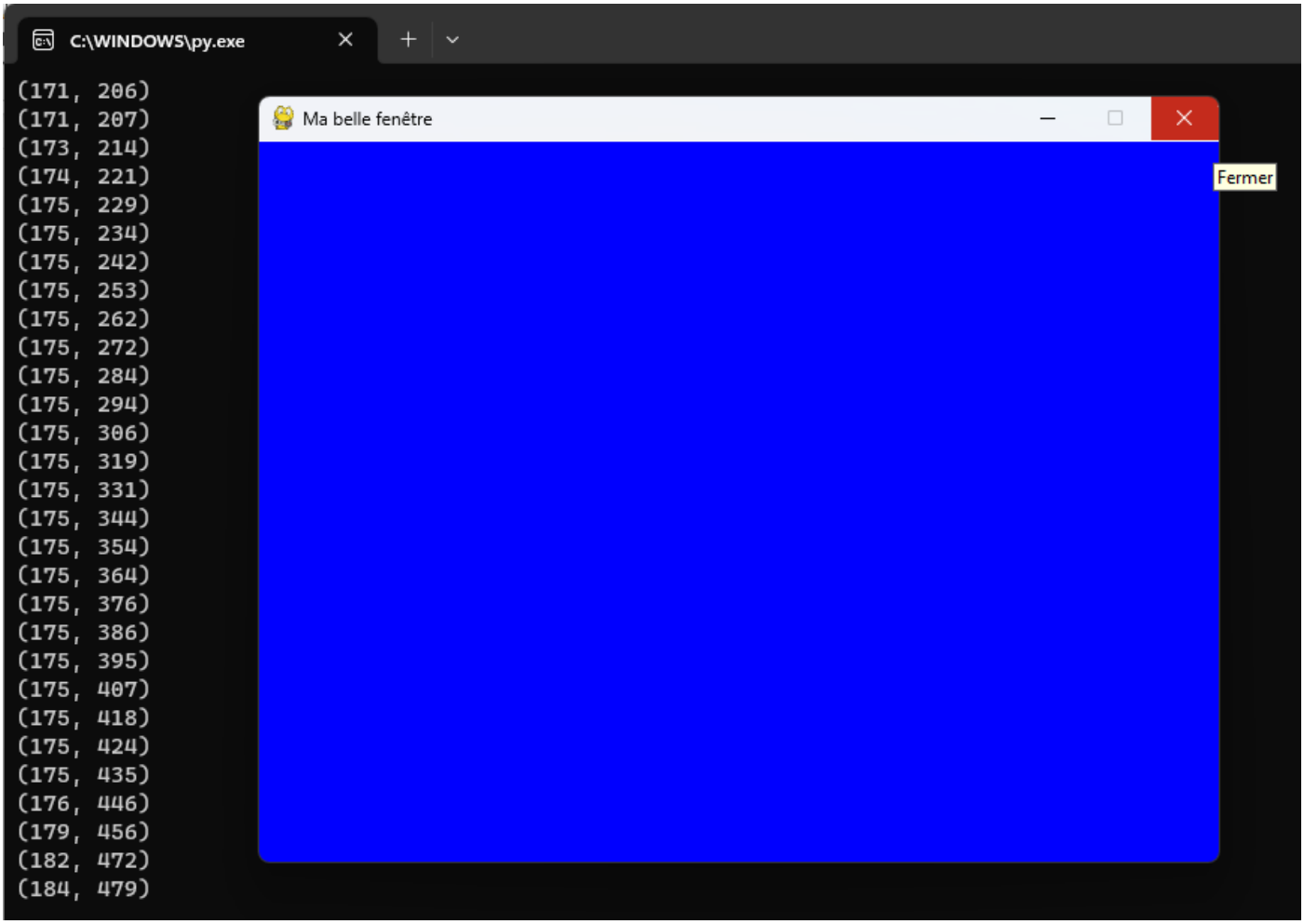
Nous allons nous intéresser à l'événement MOUSEMOTION : déplacement du curseur sur l'écran par manipulation de la souris (différent de MOUSEBUTTONUP ou MOUSEBUTTONDOWN) avec « pos » pour premier paramètre associé :
Les coordonnées du curseur seront affichées dans le terminal

```
#coding:utf-8
import pygame
#from pygame.locals import *                    #alternative
bleu=(0,0,255)
rouge=(255,0,0)
pygame.init()                                  #constructeur
resolution = (640,480)
pygame.display.set_caption("Ma belle fenêtre")
screen  =  pygame.display.set_mode(resolution)

screen.fill(bleu)                              #  pour colorier la fenêtre en bleu

pygame.display.flip()

launched = True
while launched :                    #  boucle qui permet de laisser la surface
  for event in pygame.event.get() :  # gestion des événements
#affichée
    if event.type == pygame.QUIT :
      launched= False
    elif event.type == pygame.MOUSEMOTION :
      print("{}".format(event.pos))
    screen.fill(bleu)
```


C:\WINDOWS\py.exe
(171, 206)
(171, 207)
(173, 214)
(174, 221)
(175, 229)
(175, 234)
(175, 242)
(175, 253)
(175, 262)
(175, 272)
(175, 284)
(175, 294)
(175, 306)
(175, 319)
(175, 331)
(175, 344)
(175, 354)
(175, 364)
(175, 376)
(175, 386)
(175, 395)
(175, 407)
(175, 418)
(175, 424)
(175, 435)
(176, 446)
(179, 456)
(182, 472)
(184, 479)
Ma belle fenêtre
Fermer

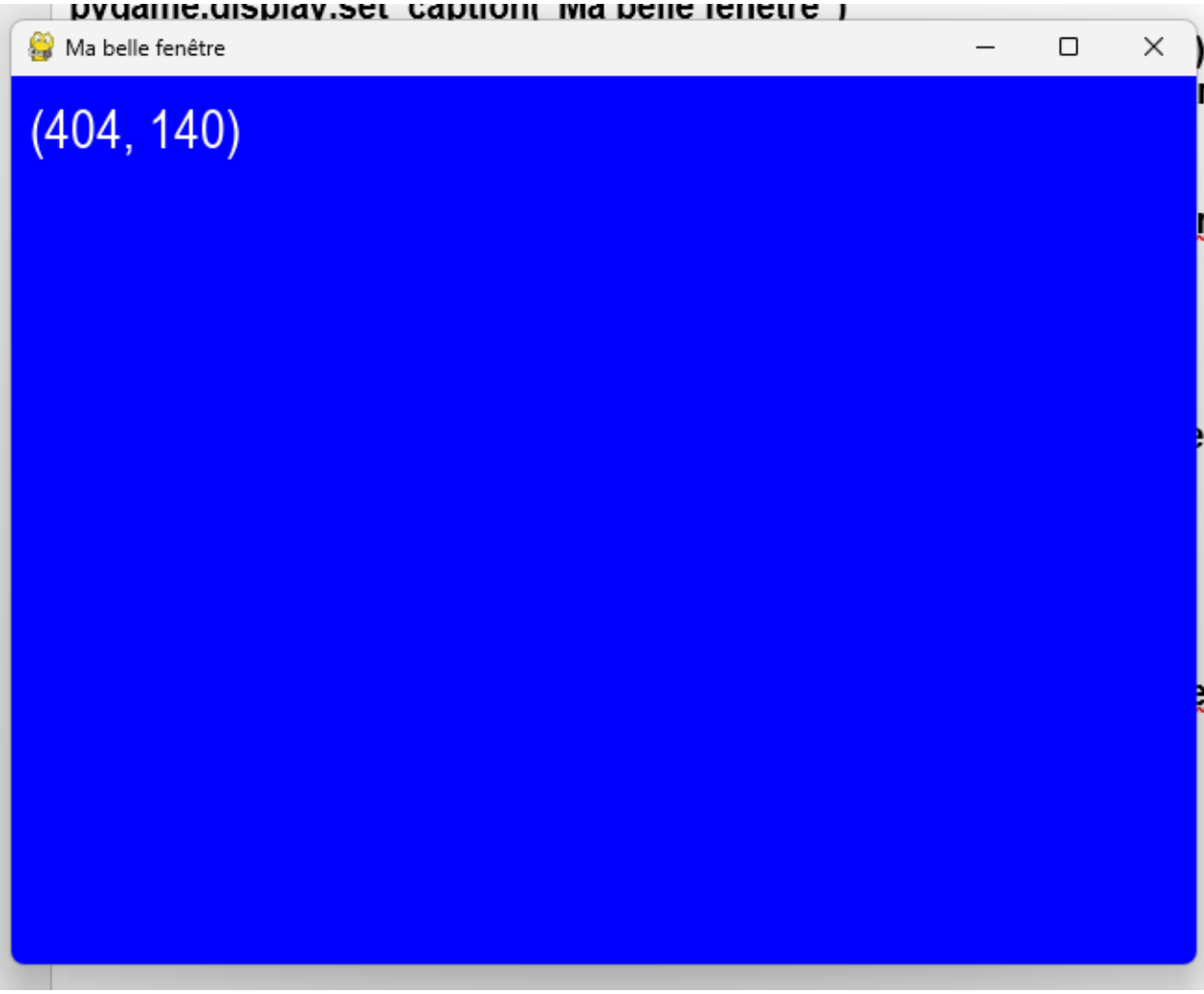
Si l'on veut afficher les coordonnées du curseur directement sur l'écran bleu, il suffit d'écrire ceci :

```
#coding:utf-8
import pygame
#from pygame.locals import *                    #alternative

pygame.init()                                   #constructeur
resolution = (640,480)
bleu    = (0,0,255)                             # creation des couleurs
blank = (255,255,255)
black  = (0,0,0)
pygame.display.set_caption("Ma belle fenêtre")
screen  =  pygame.display.set_mode(resolution,pygame.RESIZABLE)
screen.fill(bleu)                               #  pour colorier la fenêtre en bleu

arial_font= pygame.font.SysFont("arial", 30)
dimensions_text = arial_font.render("{}".format(resolution), True, blank)
screen.blit(dimensions_text, [10,10])
pygame.display.flip()

launched = True
while launched :                                #  boucle qui permet de laisser la surface
  for event in pygame.event.get() :       # gestion des événements
    if event.type == pygame.QUIT :
      launched= False
    elif event.type == pygame.MOUSEMOTION :
      screen.fill(bleu)
      dimensions_text = arial_font.render("{}".format(event.pos), True, blank)
      screen.blit(dimensions_text, [10,10])
      pygame.display.flip()
```


Ma belle fenêtre
(404, 140)