Présentation

2025/06/15 22:40

Le module **transform** de Pygame permet de créer de **nouvelles surfaces Pygame** en appliquant à une surface source une ou plusieurs transformations, isométriques ou non.

La fonction flip

flip(Surface, xbool, ybool)

La fonction flip permet de faire une symétrie axiale, verticale ou horizontale ou les deux.

Elle prend en paramètre :

- la surface que l'on va transformer,
- un booléen indiquant que l'on procède à une symétrie selon l'axe vertical en indiquant True,
- un booléen indiquant que l'on procède à une symétrie selon l'axe horizontal en indiquant True.

Si les deux booléens sont à True, alors on procède aux deux symétries.

```
IMG_FRAISE = pygame.image.load("fraise.png")
IMG_FRAISE = pygame.transform.flip(IMG_FRAISE, True, True)
```

Choisissez une image png et téléchargez-là sur le site. Comme exemple, ce sera l'image **fraise.png** qui sera utilisée.

Pour ce projet l'image :

- doit avoir les dimensions de 40×40 pixels,
- a le coin supérieur gauche positionné aux coordonnées (50, 50).

Voici les variables à définir :

```
# image FRAISE
x_fraise = 50
y_fraise = 50
largeur_fraise = 40
hauteur_fraise = 40
IMG_FRAISE = pygame.image.load("fraise.png")
```

Déplacement de l'image avec le clavier

Voici les déplacement à gérer :

- Quand on appuie sur la touche [Flèche à droite] du clavier, la fraise de déplace à droite.
- Quand on appuie sur la touche [Flèche à gauche] du clavier, la fraise de déplace à gauche

```
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RIGHT:
            x_fraise = x_fraise + 4
        elif event.key == pygame.K_LEFT:
            x_fraise = x_fraise - 4
```

Pour indiquer à Pygame d'afficher l'image en fonction des coordonnées :

```
\texttt{ECRAN.blit}(\texttt{IMG\_FRAISE}, (x\_\texttt{fraise}, y\_\texttt{fraise}))
```

Le programme complet

import pygame
pygame.init()
HAUTEUR = 400

```
LARGEUR = 400
COULEUR_FOND = (255, 255, 255)
ECRAN = pygame.display.set_mode((LARGEUR, HAUTEUR))
ARRET = False
# image FRAISE
x_fraise = 50
y_fraise = 50
largeur_fraise = 40
hauteur_fraise = 40
IMG_FRAISE = pygame.image.load("fraise.png")
while not ARRET:
  ECRAN.fill(COULEUR FOND)
  for event in pygame.event.get():
     if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_ESCAPE:
           ARRET = True
        elif event.key == pygame.K_RIGHT:
           x_fraise = x_fraise + 4
        elif event.key == pygame.K_LEFT:
           x_fraise = x_fraise - 4
  ECRAN.blit(IMG_FRAISE, (x_fraise, y_fraise))
  pygame.display.update()
```

Déplacement automatique d'une image

Cette fois -ci c'est image d'un ballon qui va se déplacer de gauche à droite. N'oubliez pas de télécharger cette image ballon.png

Voici les variables à définir :

```
# image Ballon
x_ballon = 10
y_ballon = 10
largeur_ballon = 20
hauteur_ballon = 20
IMG_BALLON = pygame.image.load("ballon.png")
```

Précisons la vitesse de déplacement du ballon sur les axes x et y .

ballonSpeed = [1, 0]

Création d'une surface invisible qui correspond aux dimensions du ballon

```
ballonRect = IMG_BALLON.get_rect()
```

En Pygame, la notion de surface est fondamentale, car la manipulation de cet élément de géométrie est un aspect important et conséquent du développement du jeu vidéo. Une surface correspond à une ligne affichée sur l'écran ou à un polygone affiché sur l'écran ; ce polygone peut être rempli de couleur, ou non. Une part importante de la gestion graphique consistera donc à créer et à manipuler les surfaces Pygame.

Pour déplacer le ballon à chaque itération, on inclut l'image du ballon dans la surface rectangle et on déplace cette surface :

```
ECRAN.blit(IMG_BALLON, ballonRect)
ballonRect = ballonRect.move(ballonSpeed)
```

Quand le ballon quitte la fenêtre, on inverse la vitesse de déplacement :

```
if ballonRect.right > LARGEUR:
    ballonSpeed[0] = - ballonSpeed[0]
```

Pour gérer la vitesse de rafraîchissement de l'écran, on peut introduite un délai en millisecondes. Par exemple un délai de 100 millisecondes fera exécuter la boucle 10 fois par seconde :

```
pygame.time.delay(100)
```

Une autre solution consiste à définir le nombre maximum d'images par seconde avec la fonction tick de l'objet Clock de Pygame :

clock = pygame.time.Clock()

dans la boucle de jeu définir un maximum de 20 images par seconde clock.tick(20)

Le programme complet

```
import pygame
pygame.init()
HAUTEUR = 400
LARGEUR = 400
COULEUR_FOND = (255, 255, 255)
ECRAN = pygame.display.set_mode((LARGEUR, HAUTEUR))
ARRET = False
# image FRAISE
x_fraise = 50
y_fraise = 50
largeur_fraise = 40
hauteur fraise = 40
IMG_FRAISE = pygame.image.load("fraise.png")
# image Ballon
x ballon = 10
y_ballon = 10
largeur_ballon = 20
hauteur_ballon = 20
IMG_BALLON = pygame.image.load("ballon.png")
ballonSpeed = [1, 0]
#IMG_BALLON.move(ballonSpeed)
ballonRect = IMG_BALLON.get_rect()
while not ARRET:
  pygame.time.delay(100)
  ECRAN.fill(COULEUR FOND)
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      sys.exit()
    elif event.type == pygame.KEYDOWN:
      if event.key == pygame.K RIGHT:
        x_fraise = x_fraise + 4
      elif event.key == pygame.K_LEFT:
        x_fraise = x_fraise - 4
  ECRAN.blit(IMG_FRAISE, (x_fraise, y_fraise))
  ECRAN.blit(IMG_BALLON, ballonRect)
  ballonRect = ballonRect.move(ballonSpeed)
  if ballonRect.right > LARGEUR:
    ballonSpeed[0] = - ballonSpeed[0]
```

pygame.display.update()

Les activités ...

Je reviens à la liste des activités.

https://repl.it/@charlestecher/Gerer-une-image <html> <iframe height="400px" width="100%" src="https://repl.it/@charlestecher/Gerer-une-image?lite=true" scrolling="no" frameborder="no" allowtransparency="true" allowfullscreen="true" sandbox="allow-forms allow-pointer-lock allow-popups allow-same-origin allow-scripts allow-modals"

I.

></iframe>

/html>

Les activités ...

Je reviens à la liste des activités.

From: / - Les cours du BTS SIO

Permanent link: /doku.php/isn/pygame03?rev=1573724317

Last update: 2019/11/14 10:38

