eBook Gratuit

APPRENEZ pygame

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#pygame

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: pygame

It is an unofficial and free pygame ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official pygame.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec pygame

Remarques

Pygame est un wrapper Python pour SDL - une bibliothèque C multi-plateforme pour contrôler le multimédia -, écrite par Pete Shinners. Cela signifie qu'en utilisant pygame, vous pouvez écrire des jeux vidéo ou d'autres applications multimédias en Python qui s'exécuteront sans aucune modification sur les plateformes prises en charge par SDL (Windows, Unix, Mac, beOS et autres).

Cette section fournit une vue d'ensemble de ce qu'est pygame et pourquoi un développeur peut vouloir l'utiliser.

Il convient également de mentionner tous les grands sujets au sein de Pygame, et d'établir un lien avec les sujets connexes. La documentation de pygame étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	==========>	Date de sortie
Pygame 1.9.0	========>	1 août 2009
Pygame 1.8.1	========>	30 juillet 2008
Pygame 1.8.0	========>	29 mars 2008
Pygame 1.7.1	========>	16 août 2005
Pygame 1.6.2	========>	-
Pygame 1.6	=========>	23 octobre 2003
Pygame 1.5	=========>	30 mai 2002
Pygame 1.4	=========>	Jan 30, 2002
Pygame 1.3	=========>	19 décembre 2001
Pygame 1.2	=========>	4 septembre 2001
Pygame 1.1	=========>	23 juin 2001
Pygame 1.0	=========>	5 avril 2001
Pygame 0.9	=========>	Feb 13, 2001
Pygame 0.5	=========>	Jan 6 14, 2001
Pygame 0.4	=========>	14 décembre 2000
Pygame 0.3	=========>	20 novembre 2000
Pygame 0.2	=========>	3 novembre 2000
Pygame 0.1	=========>	28 octobre 2000

Examples

Un simple 'jeu'

Importer et initialiser

Chaque module doit être importé et pygame ne fait pas exception. Bien que nous devions appeler la fonction `pygame.init()` pour que tous les modules importés dans pygame soient correctement initialisés. Si nous oublions cela, certains modules ne fonctionneront pas. La fonction renvoie également un tuple de toutes les initialisations réussies et échouées (cela ne provoquera pas d'erreur si un module ne parvient pas à s'initialiser).

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))
```

Créer des nécessités

Nous devons également créer un affichage. Pygame a déjà créé un affichage (caché), il suffit donc de définir le mode d'affichage (dans cet exemple, nous ne définissons que la résolution). Il est également judicieux de créer une horloge pour s'assurer que nos programmes sont mis à jour à une vitesse fixe (sinon, la vitesse serait différente selon la vitesse de l'ordinateur).

```
screen = pygame.display.set_mode((720, 480))  # Notice the tuple! It's not 2 arguments.
clock = pygame.time.Clock()
FPS = 60  # This variable will define how many frames we update per second.
```

Pour un peu de lisibilité plus tard dans notre code, nous allons créer deux constantes de couleur, représentant un tuple de rouge, vert et bleu (RVB). Les valeurs vont de 0 (pas de lumière) à 255 (pleine lumière).

```
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
```

Dans pygame, nous utilisons généralement une *surface* pour représenter l'apparence d'un objet et une *rect* (rectangle) pour représenter la position d'un objet. Une *surface* est comme une feuille de papier vierge contenant des couleurs ou des images. Si vous créez une classe, vous devez nommer les attributs *image* et *rect* car de nombreuses fonctions recherchent et utilisent ces attributs. De telles classes en tireraient avantage en héritant de la classe `pygame.sprite.Sprite` pour les raisons que vous pouvez lire ici .

```
rect = pygame.Rect((0, 0), (32, 32))  # First tuple is position, second is size.
image = pygame.Surface((32, 32))  # The tuple represent size.
image.fill(WHITE)  # We fill our surface with a nice white color (by default black).
```

La boucle du jeu

Maintenant, nous avons tout prévu pour notre boucle de jeu. Ceci est une boucle qui fonctionnera pour tout le jeu, où nous gérons les événements et met à jour l'écran et les positions de nos

objets.

D'abord, nous nous assurerons que notre boucle s'exécute à un *FPS* donné. Nous avons défini le *FPS* et créé notre horloge au début du programme. Le code suivant s'assurera que notre programme dispose de suffisamment de temps pour que notre boucle répète la quantité que nous avons définie pour notre *FPS* . Dans cet exemple, 60 fois par seconde.

```
clock.tick(FPS)
```

Ensuite, nous gérerons les événements. Un événement est essentiellement une action de l'utilisateur, telle que déplacer la souris ou appuyer sur une touche. Pygame enregistrera tous ces événements dans une file d'attente `pygame.event.get()` appelant `pygame.event.get()` . Nous pouvons itérer cela et vérifier s'il y a un événement que nous aimerions gérer. Les événements ont un attribut *type* que nous pouvons vérifier avec les constantes du module pygame pour déterminer le type d'événement.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # The user pressed the close button in the top corner of
the window.
        quit()
        # Close the program. Other methods like 'raise SystemExit' or 'sys.exit()'.
        # Calling 'pygame.quit()' won't close the program! It will just uninitialize the
modules.
```

Nous pouvons également vérifier `if event.type == pygame.KEYDOWN` pour voir si l'utilisateur a appuyé sur une touche. Dans ce cas, l'événement possède une *clé d'* attribut que nous pouvons vérifier pour voir quelle clé il représente.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        quit()
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            rect.move_ip(0, -2)  # Changes the rect's position.
        elif event.key == pygame.K_s:
            rect.move_ip(0, 2)
        elif event.key == pygame.K_a:
            rect.move_ip(-2, 0)
        elif event.key == pygame.K_d:
            rect.move_ip(2, 0)
```

Maintenant, nous devons afficher notre image. D'abord, nous pourrions vouloir effacer notre écran du rendu précédent. Nous le faisons en remplissant tout notre écran de noir (supprimez le code pour voir pourquoi nous voulons le supprimer). Ensuite, il faut *que* notre *image soit visible* à l'écran. Blitting signifie essentiellement copier l' *image* sur une autre surface (dans notre cas, l'écran). Enfin, nous *retournons* ou mettons à *jour* l'écran.

Lorsque nous sommes en train de claquer, nous n'affichons rien à l'utilisateur. Imaginez-le comme l'ordinateur d'un côté et l'utilisateur de l'autre. L'ordinateur dessine (*blits*) de son côté de l'écran, le *retourne* vers l'utilisateur, puis le répète.

```
screen.fill(BLACK)
screen.blit(image, rect)
pygame.display.update()  # Or 'pygame.display.flip()'.
```

Maintenant, nous avons un jeu de base! Assez ennuyeux, oui, mais l'essentiel est là! Combiné avec votre connaissance actuelle de Python, vous devriez pouvoir créer quelque chose de génial.

Code complet

```
import pygame
successes, failures = pygame.init()
print("{0} successes and {1} failures".format(successes, failures))


screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60  # Frames per second.

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
# RED = (255, 0, 0), GREEN = (0, 255, 0), BLUE = (0, 0, 255).

rect = pygame.Rect((0, 0), (32, 32))
image = pygame.Surface((32, 32))
image .fill(WHITE)

while True:
    clock.tick(FPS)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                rect.move_ip(0, -2)
            elif event.key == pygame.K_s:
                rect.move_ip(0, 2)
            elif event.key == pygame.K_a:
                rect.move_ip(-2, 0)
            elif event.key == pygame.K_d:
                rect.move_ip(2, 0)

    screen.fill(BLACK)
    screen.blit(image, rect)
    pygame.display.update()  # Or pygame.display.flip()
```

Mécanismes de jeu légèrement améliorés

Notez que le programme vérifie quand on appuie sur la touche et pas pour quand on appuie sur la touche. Pour résoudre ce problème, nous pourrions introduire une variable de *vitesse* . Nous pouvons créer une classe de joueur pour la maintenir plus organisée. Pour éviter le mouvement dépendant du cadre (si nous modifions le FPS à 30, les objets se déplaceraient à la moitié de la vitesse), nous introduisons un mouvement dépendant du temps en passant le temps entre les

ticks à nos objets mobiles.

```
import pygame

successes, failures = pygame.init()
print("Initializing pygame: {0} successes and {1} failures.".format(successes, failures))

screen = pygame.display.set_mode((720, 480))
clock = pygame.time.Clock()
FPS = 60

BLACK = (0, 0, 0)
WHITE = (255, 255, 255)


class Player(pygame.sprite.Sprite):
    def __init__(self):
        super().__init__()
        self.image = pygame.Surface((32, 32))
        self.image.fill(WHITE)
        self.rect = self.image.get_rect()  # Get rect of some size as 'image'.
        self.velocity = [0, 0]

    def update(self):
        self.rect.move_ip(*self.velocity)


player = Player()
running = True
while running:
    dt = clock.tick(FPS) / 1000  # Returns milliseconds between each call to 'tick'. The
convert time to seconds.
    screen.fill(BLACK)  # Fill the screen with background color.

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_w:
                player.velocity[1] = -200 * dt  # 200 pixels per second
            elif event.key == pygame.K_s:
                player.velocity[1] = 200 * dt
            elif event.key == pygame.K_a:
                player.velocity[0] = -200 * dt
            elif event.key == pygame.K_d:
                player.velocity[0] = 200 * dt
        elif event.type == pygame.KEYUP:
            if event.key == pygame.K_w or event.key == pygame.K_s:
                player.velocity[1] = 0
            elif event.key == pygame.K_a or event.key == pygame.K_d:
                player.velocity[0] = 0

    player.update()

    screen.blit(player.image, player.rect)
    pygame.display.update()  # Or pygame.display.flip()

print("Exited the game loop. Game will quit...")
quit()  # Not actually necessary since the script will exit anyway.
```

Il y a encore beaucoup de choses à améliorer concernant ce code. Je vous recommande de lire le tutoriel pygame et cette conférence de Richard Jones pour plus de détails.

Installation de pygame

Sur les fenêtres

1. Naviguez jusqu'à http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame - un site non officiel fournissant des binaires Windows de paquets python open-source pour la distribution officielle CPython par *Christoph Gohlke* .
2. Téléchargez le fichier `.whl` pygame approprié en fonction de votre version de python installée. (Le fichier s'appelle quelque chose comme `pygame - <pygame version> - <python version> - win32.whl`)
3. Courir

   ```
   pip install your-pygame-package.whl
   ```

   à l'intérieur de votre terminal, bash ou consoler.
   Remarque: si `pip` est introuvable dans `PATH` essayez d'exécuter `python -m pip install your-pygame-package.whl`
4. Vérifiez si vous pouvez importer pygame en tant que module Python

   ```
   import pygame
   ```

   Si vous ne recevez pas d'erreur, vous avez correctement installé pygame sur votre ordinateur :)

Sur linux

1. Ouvrez votre terminal et lancez

   ```
   sudo apt-get install python-pygame
   ```

   Note: Ceci installera pygame pour python2
2. Essayez d'importer pygame à l'intérieur

   ```
   import pygame
   ```

   Si vous ne recevez pas d'erreur, vous avez correctement installé pygame sur votre système Linux :)

Sur macOS

Il y a deux manières de l'installer sur mac:

Méthode 1

Accédez à la page de téléchargement Pygame et téléchargez le programme d'installation Mac. Exécutez-le et il devrait installer Pygame sur votre Mac.

Méthode 2

Installez Homebrew :

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Ensuite, utilisez Homebrew pour installer Python 2.7.12 et Pygame:

```
brew install python; brew install homebrew/python/pygame
```

Maintenant, lancez Python dans votre terminal et essayez d' `import pygame` . S'il ne dit rien, il est installé avec succès.

Importation de pygame et dessin sur un écran

Commencer

Vous devez faire ce qui suit pour commencer avec Pygame:

```
import pygame
```

Cela ouvre une fenêtre de taille 640,480 et la stocke dans une variable appelée screen.

Configurer un nom de fenêtre

La configuration d'un nom pour la fenêtre Pygame requiert la syntaxe suivante:

```
pygame.display.set_caption('Name')
```

A propos de l'écran

- Le point (0,0) se trouve dans le coin supérieur gauche de l'écran.
- les coordonnées x augmentent de gauche à droite, les coordonnées y augmentent de haut en bas. Ce sont les coordonnées du côté droit sur le plan cartésien qui sont positives et le côté gauche est négatif. en bas. (**Remarque** : Ceci est considéré si les points sont pris à partir de l'origine.)

Mise à jour de l'écran

Les modifications que vous apportez à l'écran, par exemple en le remplissant de couleur ou en y dessinant, ne s'affichent pas immédiatement!
Alors, comment le faire?
Vous devez appeler cette fonction:

```
pygame.display.update()
```

Couleurs

La coloration dans Pygame fonctionne en mode RVB.
Le code pour la coloration est:

```
color_Name = (r,g,b)
```

- R signifie rouge.
- G signifie vert
- B signifie bleu.
- Les trois devraient être des entiers compris entre 0 et 255, 255 étant le plus clair et 0 le plus sombre

Dessin

1. Dessiner des lignes

   ```
   pygame.draw.lines(screen, color, closed, pointlist, thickness)
   ```

2. Dessiner un rectangle

   ```
   pygame.draw.rect(screen, color, (x,y,width,height), thickness)
   ```

3. Dessiner un cercle

   ```
   pygame.draw.circle(screen, color, (x,y), radius, thickness)
   ```

Mettre tout en boucle

Pour faire une boucle, utilisez le code suivant:

```
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
```

```
      pygame.quit()
```

Dessiner un rectangle sur la fenêtre de pygame (code)

```
import pygame
background_colour = (255,255,255) # White color
(width, height) = (300, 200) # Screen size
color=(0,0,0) #For retangle
screen = pygame.display.set_mode((width, height)) #Setting Screen
pygame.display.set_caption('Drawing') #Window Name
screen.fill(background_colour)#Fills white to screen
pygame.draw.rect(screen, color, (100,50,30,40), 1) #Drawing the rectangle
pygame.display.update()

#Loop
running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Lire Commencer avec pygame en ligne: https://riptutorial.com/fr/pygame/topic/3959/commencer-avec-pygame

Chapitre 2: Ajout de musique de fond et d'effets sonores

Remarques

Essayez de jouer de la musique dans «.wav» au lieu de «.mp3». Dans «.mp3», la musique est à la traîne.

Examples

Exemple pour ajouter de la musique dans pygame

```
import pygame
file = 'some.mp3'
pygame.init()
pygame.mixer.init()
pygame.mixer.music.load(file)
pygame.mixer.music.play(-1) # If the loops is -1 then the music will repeat indefinitely.
```

Exemple pour ajouter une playlist de musique dans pygame

```
import pygame
import time

pygame.mixer.init()
pygame.display.init()

screen = pygame.display.set_mode ( ( 420 , 240 ) )

playlist = list()
playlist.append ( "music3.mp3" )
playlist.append ( "music2.mp3" )
playlist.append ( "music1.mp3" )

pygame.mixer.music.load ( playlist.pop() )  # Get the first track from the playlist
pygame.mixer.music.queue ( playlist.pop() ) # Queue the 2nd song
pygame.mixer.music.set_endevent ( pygame.USEREVENT )    # Setup the end track event
pygame.mixer.music.play()           # Play the music

running = True
while running:
   for event in pygame.event.get():
      if event.type == pygame.USEREVENT:    # A track has ended
         if len ( playlist ) > 0:       # If there are more tracks in the queue...
            pygame.mixer.music.queue ( playlist.pop() ) # Q
```

Lire Ajout de musique de fond et d'effets sonores en ligne:
https://riptutorial.com/fr/pygame/topic/7419/ajout-de-musique-de-fond-et-d-effets-sonores

Chapitre 3: Créer une fenêtre dans pygame - pygame.display.set_mode ()

Syntaxe

- pygame.display.set_mode (résolution = (0,0), flags = 0, depth = 0) # Renvoie un pygame.Surface représentant la fenêtre à l'écran
- flags = pygame.FULLSCREEN | pygame.OPENGL # Les drapeaux peuvent être combinés en utilisant le "|" (bitwise OR ou "pipe") caractère.

Paramètres

paramètre	explication
résolution	une paire de chiffres représentant la largeur et la hauteur de la fenêtre
drapeaux	options supplémentaires qui changent le type de fenêtre - voir "Remarques" pour les drapeaux disponibles
profondeur	quantité de bits utilisés pour la couleur

Remarques

- Les valeurs possibles pour les arguments d' `flag` sont les suivantes:

drapeau	la description
pygame.FULLSCREEN	la fenêtre est en plein écran
pygame.RESIZABLE	la fenêtre est redimensionnable
pygame.NOFRAME	la fenêtre n'a pas de bordure ou de contrôles
pygame.DOUBLEBUF	utiliser un double tampon - recommandé pour `HWSURFACE` ou `OPENGL`
pygame.HWSURFACE	la fenêtre est accélérée par le matériel, uniquement possible en combinaison avec `FULLSCREEN`
pygame.OPENGL	la fenêtre est rendu par OpenGL

D'autres remarques:

- Pygame ne peut actuellement gérer qu'une seule fenêtre à la fois. Créer une deuxième fenêtre en appelant `pygame.display.set_mode((x,y))` une seconde fois fermera la première

fenêtre.

- Changer l'argument des `depths` n'est presque jamais nécessaire - pygame choisira le meilleur par lui-même. Si une profondeur non prise en charge par le système est définie, pygame émulera cette profondeur, qui peut être très lente.
- Les éléments dessinés sur la surface renvoyée par `pygame.display.set_mode()` ne sont pas immédiatement visibles à l'écran - l'affichage doit d'abord être `pygame.display.update()` aide de `pygame.display.update()` ou `pygame.display.flip()` .

Examples

Créer une fenêtre Pygame

Cela crée une fenêtre en plein écran avec une taille de 500x500 pixels:

```
pygame.init()
screen = pygame.display.set_mode((500, 500), pygame.FULLSCREEN)
```

`screen` représente désormais la fenêtre à l'écran; c'est un objet pygame.Surface. Tout ce qui devrait être visible par l'utilisateur doit être dessiné avec `screen.blit` .

Lire Créer une fenêtre dans pygame - pygame.display.set_mode () en ligne: https://riptutorial.com/fr/pygame/topic/6442/creer-une-fenetre-dans-pygame---pygame-display-set-mode---

Chapitre 4: Créer une fenêtre Pygame

Remarques

Si vous voulez avoir d'autres couleurs comme arrière-plan, nommez une nouvelle variable telle que `red = (255,0,0)` et changez le `display.fill(black)` pour `display.fill(red)` . Vous pouvez créer des couleurs en les stockant dans une variable et en vérifiant leurs valeurs RVB à partir d'Internet.

Examples

Création de la fenêtre Pygame

```
import pygame

background_colour = (255,255,255) # For the background color of your window
(width, height) = (300, 200) # Dimension of the window

screen = pygame.display.set_mode((width, height)) # Making of the screen
pygame.display.set_caption('Tutorial 1') # Name for the window
screen.fill(background_colour) #This syntax fills the background colour

pygame.display.flip()

running = True
while running:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      running = False
      pygame.quit()
```

Lire Créer une fenêtre Pygame en ligne: https://riptutorial.com/fr/pygame/topic/6477/creer-une-fenetre-pygame

Chapitre 5: Créer une fenêtre simple de pygame

Examples

Le code complet

```
import pygame

pygame.init()

WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption('My Game')

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)

SCREEN.fill(RED)
pygame.display.flip()

is_running = True
while is_running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            is_running = False

pygame.quit()
```

Importation et initialisation de pygame

Comme nous le faisons avec tout module en python, nous devons importer pygame:

```
import pygame
```

Nous initialisons ensuite tous les modules pygame importés:

```
pygame.init()
```

Ceci est utilisé pour initialiser tous les modules pygame. Sans cela, les modules ne fonctionneraient pas

Définition des constantes

Nous définissons ensuite des constantes ici:

```
WIDTH = 300
HEIGHT = 200
SCREEN = pygame.display.set_mode((WIDTH, HEIGHT))
```

Les constantes `WIDTH` et `HEIGHT` permettent de créer une fenêtre de 300 pixels de largeur et de 200 pixels de hauteur. La fonction utilisée dans `SCREEN` , `pygame.display.set_mode((WIDTH, HEIGHT))` , définit le mode d'affichage et renvoie un objet Surface . Notez comment les paramètres de cette fonction sont les constantes `WIDTH` et `HEIGHT` définies précédemment.

Définition du nom de la fenêtre

Nous utilisons ensuite cette fonction pour changer le nom de la fenêtre en My Game:

```
pygame.display.set_caption('My Game')
```

Définir les couleurs

Nous définissons ensuite 6 couleurs utilisables dans notre fenêtre:

```
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 255)
```

Lors de la définition des couleurs, nous avons mis 3 valeurs comprises entre 0 et 255. La classe pygame.Color utilise normalement ce format:

```
COLOUR = (r, g, b, a)
```

Lorsque le paramètre r définit la valeur rouge de la couleur, le paramètre g définit la valeur verte de la couleur et le paramètre b définit la valeur bleue de la couleur. Le paramètre a définit la valeur alpha de la couleur.

Nous donnons alors cette commande:

```
SCREEN.fill(RED)
```

Ceci est une fonction pygame.Surface.fill qui remplit l'objet Surface, notre écran, avec la couleur rouge.

Utiliser pygame.display.flip ()

Nous utilisons alors cette fonction

```
pygame.display.flip()
```

Cela rend fondamentalement tout ce que nous avons dessiné à l'écran Surface visible et met à jour le contenu de l'affichage entier. Sans cette ligne, l'utilisateur ne verrait rien sur son écran pygame.

La boucle du jeu

Les prochaines lignes sont ce qu'on appelle une "boucle de jeu".

Pour commencer, nous créons une variable et la rendons vraie:

```
is_running = True
```

Pour que nous puissions commencer notre boucle while:

```
while is_running:
```

qui courra tout au long du jeu.

Dans sa forme la plus élémentaire, pygame a des "événements" qui prennent les entrées de l'utilisateur, par exemple une pression sur un bouton ou un clic de souris. Pygame gère ces événements via une file d'attente d'événements. Nous pouvons obtenir ces événements de la file d'attente des événements avec ceci pour la boucle:

```
for event in pygame.event.get():
```

Ce qui passe essentiellement par une liste d'événements, notre file d'attente d'événements. Ce sont les 2 lignes suivantes:

```
if event.type == pygame.QUIT:
    is_running = False
```

Cela fera en sorte que lorsque l'utilisateur appuie sur le bouton de sortie dans le coin supérieur, l'événement avec le type `pygame.QUIT` se produit.

Cela termine la boucle while, car `is_running` est maintenant `False` et le script passe à la ligne finale:

```
pygame.quit()
```

Ce qui désinitialise les modules pygame.

Lire Créer une fenêtre simple de pygame en ligne:
https://riptutorial.com/fr/pygame/topic/6597/creer-une-fenetre-simple-de-pygame

Chapitre 6: Dessin sur l'écran

Examples

dessiner des formes, du texte et des images sur l'écran avec une petite animation

Ce programme va dessiner des formes sur l'écran, dessinez "Bonjour tout le monde!" au milieu de l'écran et laisser une image aller à chaque coin de la fenêtre. Vous pouvez utiliser chaque image de votre **choix** , mais **vous devrez placer le fichier image dans le même répertoire que votre programme.**

le code entier:

```
import pygame, sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((400, 300), 0, 32)
pygame.display.set_caption('animation')

#set up the colors
white = (255, 255, 255)
black = (  0,   0,   0)
green = (0, 255, 0)
blue = (0, 0, 180)
red   = (255,   0,   0)

image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'

# text setting
font_obj = pygame.font.Font('freesansbold.ttf', 32)
text_surface_obj = font_obj.render('Hello World!', True, GREEN, BLUE)
text_rect_obj = text_surface_obj.get_rect()
text_rectObj.center = (200, 150)

while True: # the main game loop
    screen.fill(WHITE)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))

    # draw some blue lines onto the surface
```

```
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)

    # draw a blue circle onto the surface
    pygame.draw.circle(screen, blue, (300, 50), 20, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text_surface_obj, text_rect_obj)


    #the animation of the image
    if direction == 'right':
        imagex += 5
        if imagex == 360:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 260:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
            direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
            direction = 'right'
    screen.blit(image, (imagex, imagey))

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    pygame.display.update()
    fpsClock.tick(FPS)
```

dessiner le fond blanc:

```
screen.fill(white)
```

dessiner le polygone:

Dans cette fonction, vous définissez la surface d'affichage, la couleur et la position de chaque coin du polygone, vous pouvez le faire dans le sens des aiguilles d'une montre et dans le sens inverse.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

dessiner les lignes:

Vous définissez ici la surface d'affichage, la couleur, le premier et le dernier point et la largeur de la ligne.

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

dessiner le cercle:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position, le rayon et la largeur du cercle (0 donne un cercle simple).

```
pygame.draw.circle(screen, blue, (300, 50), 20, 0)
```

dessiner l'ellipse:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position, la taille horizontale, la taille verticale et la largeur de l'ellipse.

```
pygame.draw.ellipse(screen, red, (100, 150, 40,80), 1)
```

dessiner le rectangle:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position et la verticale et la taille horizontale du rectangle.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

définir le texte:

Tout d'abord, vous définissez le type et la taille de votre texte, j'utilise une police de base que vous obtenez avec pygame.

```
font_obj = pygame.font.Font('freesansbold.ttf', 32)
```

Vous définissez ensuite le texte réel, si vous le souhaitez en gras ou non (True / False), la couleur du texte et, si vous souhaitez marquer votre texte, une couleur du marquage.

```
text_surface_obj = font_obj.render('Hello World!', True, green, blue)
```

Si vous souhaitez marquer votre texte ou définir le centre de votre texte, vous devez indiquer à pygame que cette fonction:

```
text_rect_obj = text_surface_obj.get_rect()
```

Et après cela, vous pouvez définir le centre de votre texte avec cette fonction:

```
text_rect_obj.center = (200, 150)
```

dessiner le texte:

Si vous avez marqué votre texte ou défini le centre, vous devez dessiner le texte de cette manière:

```
screen.blit(text_surface_obj, text_rectObj)
```

Sinon, vous dessinez votre texte, mais vous devez définir la position, vous le faites ainsi:

```
DISPLAYSURF.blit(textSurfaceObj, (100,50))
```

définir l'image:

Vous définissez ici l'image que vous souhaitez utiliser, la position de départ (coordonnées x et y) et la direction de l'image.

```
image  = pygame.image.load('image.png')
imagex = 360
imagey = 260
direction = 'left'
```

animer l'image:

Ici, vous contrôlez la direction de l'image, si elle a atteint un coin, si c'est le cas, changez de direction, sinon déplacez-la de 5 pixels dans la même direction et dessinez à nouveau l'image. C'est ce que nous faisons avec cette partie du code:

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
```

```
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
    imagex -= 5
    if imagex == 20:
        direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
        direction = 'right'
screen.blit(image, (imagex, imagey))
```

note: mon image est de 20 par 20 pixels, j'ai utilisé `if imagex == 360` et `if imagey == 260:` car alors mon image est à 20 pixels du bord, si votre image a une taille différente, vous devrez changer les chiffres .

vérifier si vous quittez le programme:

Ici, nous vérifions si vous avez fermé la fenêtre de votre programme.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

mettre à jour l'affichage:

Ici, vous indiquez à pygame de mettre à jour l'affichage afin que tout ce que vous avez dessiné apparaisse à l'écran.

```
pygame.display.update()
```

définir les images par seconde:

Ici, vous dites à pygame de dormir suffisamment pour que le réglage des images par seconde soit respecté.

```
fpsClock.tick(FPS)
```

Lire Dessin sur l'écran en ligne: https://riptutorial.com/fr/pygame/topic/6287/dessin-sur-l-ecran

Chapitre 7: Dessin sur l'écran

Syntaxe

- pygame.draw.rect (Surface, couleur, Rect, width = 0)
- pygame.draw.polygon (Surface, couleur, liste de points, largeur = 0)
- pygame.draw.circle (Surface, couleur, pos, rayon, largeur = 0)
- pygame.draw.ellipse (Surface, couleur, Rect, width = 0)
- pygame.draw.arc (Surface, couleur, Rect, start_angle, stop_angle, width = 1)
- pygame.draw.line (Surface, couleur, start_pos, end_pos, width = 1)
- pygame.draw.lines (Surface, couleur, fermé, liste de points, largeur = 1)
- pygame.draw.aaline (Surface, couleur, startpos, endpos, blend = 1)
- pygame.draw.aalines (Surface, couleur, fermé, liste de points, mélange = 1)

Paramètres

Paramètres	Détails
Surface	La surface sur laquelle dessiner la forme.
Couleur	Une séquence de 3 ou 4 nombres entiers représentant le rouge, le vert et le bleu (et l'alpha), chaque valeur variant entre 0 et 255.
Rect	Une zone rectangulaire où la forme sera attirée.
largeur	La largeur des lignes. La forme sera remplie si width = 0.
liste de points	Une liste d'une quantité arbitraire de points / sommets, en pixels (x, y).
pos	La position du centre du cercle, en pixels (x, y).
rayon	Le rayon des cercles en pixels.
fermé	Si true, une ligne entre le dernier et le premier point sera dessinée, fermant la forme.
mélange = 1	Si la valeur est true, les nuances seront mélangées avec les nuances de pixels existantes au lieu de les écraser.
start_angle	L'angle initial de l'arc, en radians.
stop_angle	L'angle final de l'arc, en radians.
start_pos	La position de départ de la ligne, en pixels.
end_pos	La position de fin de la ligne, en pixels

Examples

Dessin avec le module de dessin

Pygame a un module, `pygame.draw` , qui contient des fonctions qui peuvent dessiner des formes directement sur une surface.

Fonction	La description
pygame.draw.rect	dessiner un rectangle
pygame.draw.polygon	dessiner une forme avec n'importe quel nombre de côtés
pygame.draw.circle	dessine un cercle autour d'un point
pygame.draw.ellipse	dessiner une forme ronde à l'intérieur d'un rectangle
pygame.draw.arc	dessiner une section partielle d'une ellipse
pygame.draw.line	dessiner un segment de droite
pygame.draw.lines	dessiner plusieurs segments de ligne contigus
pygame.draw.aaline	dessiner de fines lignes anti-aliasées
pygame.draw.aalines	dessiner une séquence connectée de lignes antialiasées

Comment utiliser le module

Pour utiliser le module, vous devez d'abord importer et initialiser correctement pygame et définir un mode d'affichage. Il est pratique de définir les constantes de couleur à l'avance, ce qui rend votre code plus lisible et plus beau. Toutes les fonctions prennent une surface à dessiner, une couleur et un argument de position qui sont soit un pygame Rect, soit une séquence entière / flottante à 2 éléments (le `pygame.draw.circle` ne prendra que des entiers pour des raisons indéfinies).

Exemple

Le code ci-dessous présentera toutes les différentes fonctions, leur utilisation et leur apparence. Nous allons initialiser pygame et définir des constantes avant les exemples.

```
import pygame
from math import pi
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)
```

La couleur noire est la couleur par défaut de la surface et représente la partie de la surface sur laquelle on n'a pas dessiné. Les paramètres de chaque fonction sont expliqués ci-dessous dans **Paramètres** .

Rect

```
size = (50, 50)

rect_border = pygame.Surface(size)  # Create a Surface to draw on.
pygame.draw.rect(rect_border, RED, rect_border.get_rect(), 10)  # Draw on it.

rect_filled = pygame.Surface(size)
pygame.draw.rect(rect_filled, RED, rect_filled.get_rect())
```

Polygone

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]  # The corner points of the polygon.

polygon = pygame.Surface(size)
pygame.draw.polygon(polygon, RED, points, 10)

polygon_filled = pygame.Surface(size)
pygame.draw.polygon(polygon_filled, RED, points)
```

Cercle

```
size = (50, 50)
radius = 25

circle = pygame.Surface(size)
pygame.draw.circle(circle, RED, (radius, radius), radius, 10)  # Position is the center of the
circle.

circle_filled = pygame.Surface(size)
pygame.draw.circle(circle_filled, RED, (radius, radius), radius)
```

Les trous sont une conséquence malheureuse de l'algorithme de dessin de pygame.

Ellipse

```
size = (50, 25)  # Minimize it's height so it doesn't look like a circle.

ellipse = pygame.Surface(size)
pygame.draw.ellipse(ellipse, RED, ellipse.get_rect(), 5)

ellipse_filled = pygame.Surface(size)
pygame.draw.ellipse(ellipse_filled, RED, ellipse.get_rect())
```

Les trous sont une conséquence malheureuse de l'algorithme de dessin de pygame.

Arc

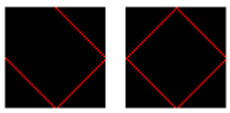
```
size = (50, 50)

arc = pygame.Surface(size)
pygame.draw.arc(arc, RED, arc.get_rect(), 0, pi)  # 0 to pi is 180° creating a half circle.
```

Ligne

```
size = (50, 50)

line = pygame.Surface(size)
pygame.draw.line(line, RED, (0, 0), (50, 50))  # Start at topleft and ends at bottomright.
```

Lignes

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

lines = pygame.Surface(size)
pygame.draw.lines(lines, RED, False, points)

lines_closed = pygame.Surface(size)
pygame.draw.lines(lines_closed, RED, True, points)
```

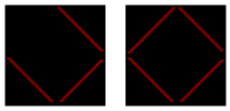


Ligne anti-aliasée

```
size = (50, 50)

antialiased_line = pygame.Surface(size)
pygame.draw.aaline(antialiased_line, RED, (0, 0), (50, 50))
```

Lignes anti-aliasées

```
size = (50, 50)
points = [(25, 0), (50, 25), (25, 50), (0, 25)]

antialiased_lines = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines, RED, False, points)

antialiased_lines_closed = pygame.Surface(size)
pygame.draw.aalines(antialiased_lines_closed, RED, True, points)
```



Essaye le

Pour l'essayer vous-même: copiez l'un des extraits de code ci-dessus et le code ci-dessous dans un fichier vide, remplacez l' *image* du nom par le nom de la surface que vous souhaitez rendre visible et expérimentez.

```
import pygame
from math import pi
pygame.init()

screen = pygame.display.set_mode((100, 100))
WHITE = pygame.Color(255, 255, 255)
RED = pygame.Color(255, 0, 0)

# But code snippet here

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()

    screen.blit(image, (25, 25))
    pygame.display.update()
```

Les surfaces

Dans Pygame, vous utilisez généralement les surfaces pour représenter l'apparence des objets et les rectangles pour représenter leurs positions. Une surface est comme une feuille de papier vierge contenant des couleurs ou des images. Il y a deux manières de créer une surface: vide à partir de zéro ou en chargeant une image.

Créer une surface

Pour créer une surface, il vous faut au minimum sa taille, qui est une séquence de 2 éléments de largeur et de hauteur, représentant la taille en pixels.

Vous pouvez également transmettre des arguments supplémentaires lors de la création d'une surface pour contrôler la profondeur de bits, les masques et les fonctionnalités supplémentaires en tant qu'alpha par pixel et / ou créer l'image dans la mémoire vidéo. Cela sort cependant du cadre de cet exemple.

```
size = width, height = (32, 32)
empty_surface = pygame.Surface(size)
```

Vous pouvez utiliser le module `pygame.draw` pour dessiner des formes sur la surface ou le remplir avec une couleur en appelant le `fill(color)` la méthode Surface. La *couleur de l'* argument est une séquence entière de 3 ou 4 éléments ou un objet `pygame.Color` .

Charger une image

Le plus souvent, vous souhaitez utiliser vos propres images dans un jeu (appelées sprites). Créer une surface avec votre image est aussi simple que:

```
my_image = pygame.image.load(path_to_image)
```

Le chemin vers l'image peut être relatif ou absolu. Pour améliorer les performances, il est généralement judicieux de convertir votre image au même format de pixel que l'écran. Cela peut être fait en appelant la méthode de `convert()` , comme ceci:

```
my_image = pygame.image.load(path_to_image).convert()
```

Si votre image contient de la transparence (valeurs alpha), il vous suffit d'appeler la méthode `convert_alpha()` place:

```
my_image = pygame.image.load(path_to_image).convert_alpha()
```

Blitting

Les surfaces doivent être transparentes à l'écran pour pouvoir les afficher. Blitting signifie essentiellement copier des pixels d'une Surface à une autre (l'écran est également une Surface). Vous devez également passer la position de la surface, qui doit être une séquence entière à 2

éléments ou un objet Rect. La partie supérieure de la surface sera placée à la position.

```
screen.blit(my_image, (0, 0))
pygame.display.update()  # or pygame.display.flip()
```

Il est possible de blinder d'autres surfaces que l'écran. Pour afficher ce qui a été affiché à l'écran, vous devez appeler `pygame.display.update()` ou `pygame.display.flip()` .

Transparence

Il existe trois types de transparence pris en charge dans pygame: colorkeys, alphas de surface et alphas par pixel.

Colorkeys

Donne une couleur totalement transparente, ou plus exactement, en faisant simplement une couleur qui ne soit pas blit. Si vous avez une image avec un rectangle noir à l'intérieur, vous pouvez définir un colorkey pour empêcher la couleur noire d'être blit.

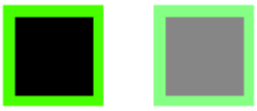
```
BLACK = (0, 0, 0)
my_image.set_colorkey(BLACK)  # Black colors will not be blit.
```

Une surface ne peut avoir qu'un seul colorkey. Définir un autre colorkey remplacera le précédent. Les Colorkeys ne peuvent pas avoir de valeurs alpha différentes, elles ne peuvent que rendre une couleur invisible.

Alphas de surface

Rend toute la surface transparente par une valeur alpha. Avec cette méthode, vous pouvez avoir différentes valeurs alpha mais cela affectera toute la surface.

```
my_image.set_alpha(100)  # 0 is fully transparent and 255 fully opaque.
```



Alpha par pixel

Rend chaque pixel de la Surface transparent par une valeur alpha individuelle. Cela vous donne le plus de liberté et de flexibilité, mais c'est aussi la méthode la plus lente. Cette méthode nécessite également que la surface soit créée en tant que surface alpha par pixel et que les arguments de couleur doivent contenir un quatrième entier alpha.

```
size = width, height = (32, 32)
my_image = pygame.Surface(size, pygame.SRCALPHA)  # Creates an empty per-pixel alpha Surface.
```

La surface va maintenant dessiner la transparence si la couleur contient la quatrième valeur alpha.

```
BLUE = (0, 0, 255, 255)
pygame.draw.rect(my_image, BLUE, my_image.get_rect(), 10)
```

Contrairement aux autres surfaces, cette couleur par défaut de la surface ne sera pas noire mais transparente. C'est pourquoi le rectangle noir au milieu disparaît.

Combinaison de colorkey et de surface alpha

Colorkeys et Surface alphas peuvent être combinés, mais alpha par pixel impossible. Cela peut être utile si vous ne voulez pas les performances plus lentes d'une surface par pixel.

```
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)
```

Code complet

Copiez ceci dans un fichier vide et lancez-le. Appuyez sur les touches 1, 2, 3 ou 4 pour faire apparaître les images. Appuyez sur 2, 3 ou 4 fois pour les rendre plus opaques.

```
import pygame
pygame.init()

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255, 50)  # This color contains an extra integer. It's the alpha value.
PURPLE = (255, 0, 255)

screen = pygame.display.set_mode((200, 325))
screen.fill(WHITE)  # Make the background white. Remember that the screen is a Surface!
clock = pygame.time.Clock()

size = (50, 50)
red_image = pygame.Surface(size)
green_image = pygame.Surface(size)
blue_image = pygame.Surface(size, pygame.SRCALPHA)  # Contains a flag telling pygame that the
Surface is per-pixel alpha
```

```
purple_image = pygame.Surface(size)

red_image.set_colorkey(BLACK)
green_image.set_alpha(50)
# For the 'blue_image' it's the alpha value of the color that's been drawn to each pixel that
determines transparency.
purple_image.set_colorkey(BLACK)
purple_image.set_alpha(50)

pygame.draw.rect(red_image, RED, red_image.get_rect(), 10)
pygame.draw.rect(green_image, GREEN, green_image.get_rect(), 10)
pygame.draw.rect(blue_image, BLUE, blue_image.get_rect(), 10)
pygame.draw.rect(purple_image, PURPLE, purple_image.get_rect(), 10)

while True:
    clock.tick(60)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quit()
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_1:
                screen.blit(red_image, (75, 25))
            elif event.key == pygame.K_2:
                screen.blit(green_image, (75, 100))
            elif event.key == pygame.K_3:
                screen.blit(blue_image, (75, 175))
            elif event.key == pygame.K_4:
                screen.blit(purple_image, (75, 250))

    pygame.display.update()
```

Lire Dessin sur l'écran en ligne: https://riptutorial.com/fr/pygame/topic/7079/dessin-sur-l-ecran

Chapitre 8: Gestion des événements

Examples

Boucle d'événement

Pygame enregistrera tous les événements de l'utilisateur dans une file d'attente d'événements pouvant être reçue avec le code `pygame.event.get()` . Chaque élément de cette file d'attente est un objet `Event` et ils auront tous le `type` attribut, qui est un entier représentant le type d'événement. Dans le module pygame, il existe des constantes entières prédéfinies représentant le type. À l'exception de cet attribut, les événements ont des attributs différents.

Nom constant	Les attributs
QUITTER	aucun
ACTIVEEVENT	gain, état
TOUCHE BAS	unicode, clé, mod
KEYUP	clé, mod
MOUSEMOTION	pos, rel, boutons
MOUSEBUTTONUP	pos, bouton
MOUSEBUTTONDOWN	pos, bouton
JOYAXISMOTION	joie, axe, valeur
JOYBALLMOTION	joie, bal, rel
JOYHATMOTION	joie, chapeau, valeur
JOYBUTTONUP	joie, bouton
JOYBUTTONDOWN	joie, bouton
VIDEORESIZE	taille, w, h
VIDEOEXPOSE	aucun
USEREVENT	code

Exemple

Pour gérer nos événements, nous parcourons simplement la file d'attente, vérifions le type (à

l'aide des constantes prédéfinies dans le module pygame) et exécutons une action. Ce code vérifie si l'utilisateur a appuyé sur le bouton de fermeture dans le coin supérieur de l'écran et, si tel est le cas, termine le programme.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        # Close the program any way you want, or troll users who want to close your program.
        raise SystemExit
```

ATTENTION : vous devez appeler la file d'attente des événements régulièrement lorsque vous utilisez pygame! Outre la récupération des événements disponibles, l'appel de la file d'attente d'événements est également la manière dont pygame peut interagir avec le système d'exploitation en interne. Si la file d'attente des événements n'est pas appelée régulièrement, votre système d'exploitation supposera que votre programme ne fonctionnera plus correctement et fera peut-être croire que le programme est bloqué (dans Windows, la fenêtre devient blanche). Si vous ne voulez rien faire avec les événements, vous devez appeler `pygame.event.pump()` chaque boucle de jeu pour que pygame traite les événements en interne.

Événements clavier

Il existe deux types d'événements clés dans pygame: `KEYDOWN` et `KEYUP` . Ces événements ont une `key` attribut qui est un entier représentant une touche du clavier. Le module pygame possède des constantes entières prédéfinies représentant toutes les clés communes. Les constantes sont nommées avec un `K` majuscule, un trait de soulignement et le nom de la clé. Par exemple, `<-` s'appelle `K_BACKSPACE` , `a` s'appelle `K_a` et `F4` s'appelle `K_F4` .

Exemple

Ce code vérifiera si l'utilisateur a appuyé sur `w` , `a` , `s` ou `d` .

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # Usually wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_w:
            print("Player moved up!")
        elif event.key == pygame.K_a:
            print("Player moved left!")
        elif event.key == pygame.K_s:
            print("Player moved down!")
        elif event.key == pygame.K_d:
            print("Player moved right!")
```

Modificateurs

Il n'y a pas de constante entière pour les majuscules. Au lieu de cela, les événements clés ont un autre attribut appelé `mod` , qui consiste à appuyer simultanément sur les modificateurs (`shift` , `ctrl`

, `alt`, etc.) en tant que clé. L'attribut `mod` est un entier représentant le modificateur pressé. La valeur entière de chaque modificateur est stockée dans le module pygame sous le nom de `KMOD_` et son nom. Par exemple, `Left shift` est nommé `KMOD_LSHIFT` , `Tab` est nommé `KMOD_TAB` et `Ctrl` est nommé `KMOD_CTRL` .

Exemple

Ce code vérifie si l'utilisateur a appuyé sur `a` , `Maj gauche` + `a` ou `Caps` + `a` .

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:  # It's still wise to be able to close your program.
        raise SystemExit
    elif event.type == pygame.KEYDOWN:
        if event.key == pygame.K_a:
            if event.mod == 0:  # No modifier.
                print("You pressed 'a'")
            elif event.mod == pygame.KMOD_LSHIFT or event.mod == pygame.KMOD_CAPS:
                print("You pressed 'A'")
            else:
                print("You pressed 'a' with another modifier than right shift or caps.")
```

Événements souris

Il existe trois types d'événements de souris dans pygame `MOUSEMOTION` , `MOUSEBUTTONDOWN` et `MOUSEBUTTONUP` . Pygame enregistrera ces événements lorsqu'un mode d'affichage a été défini.

`MOUSEMOTION` est reçu lorsque l'utilisateur déplace sa souris sur l'écran. Il a les `buttons` attributs, `pos` et `rel` .

- *`buttons`* est un tuple représentant si les boutons de la souris (`left` , `mouse-wheel` , `right`) sont pressés ou non.
- *`pos`* est la position absolue (`x` , `y`) du curseur en pixels.
- *`rel`* est la position relative à la position précédente (`rel_x` , `rel_y`) en pixels.

`MOUSEBUTTONDOWN` et **`MOUSEBUTTONUP`** sont reçus lorsque l'utilisateur appuie ou relâche un bouton de la souris. Ils ont le `button` attributs et `pos` .

- *`button`* est un entier représentant le bouton pressé. `1` pour le bouton gauche, 2 pour la molette et `3` pour le bouton droit.
- *`pos`* est la position absolue de la souris (`x` , `y`) lorsque l'utilisateur appuie sur le bouton de la souris.

Exemple

Voici un court exemple utilisant certains des attributs de chaque événement de souris:

```
for event in pygame.event.get():
```

```
    if event.type == pygame.QUIT:  # Close your program if the user wants to quit.
        raise SystemExit
    elif event.type == pygame.MOUSEMOTION:
        if event.rel[0] > 0:  # 'rel' is a tuple (x, y). 'rel[0]' is the x-value.
            print("You're moving the mouse to the right")
        elif event.rel[1] > 0:  # pygame start y=0 at the top of the display, so higher y-
values are further down.
            print("You're moving the mouse down")
    elif event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            print("You pressed the left mouse button")
        elif event.button == 3:
            print("You pressed the right mouse button")
    elif event.type == pygame.MOUSEBUTTONUP:
        print("You released the mouse button")
```

Comme il n'y a pas de constantes prédéfinies pour l'attribut du bouton de la souris dans le module pygame, voici les valeurs pour chacun:

Bouton	Valeur
Bouton gauche de la souris	1
Bouton de la molette de la souris	2
Bouton droit de la souris	3
Molette de la souris défiler vers le haut	4
Molette de la souris défiler vers le bas	5

`pygame.MOUSEBUTTONDOWN` `pygame.MOUSEBUTTONUP` le bouton de la souris `pygame.MOUSEBUTTONUP` événements `pygame.MOUSEBUTTONDOWN` et `pygame.MOUSEBUTTONUP` .

Vérification d'état

Il est possible d'appeler des fonctions du module `pygame.key` et `pygame.mouse` pour recevoir l'état de la clé et de la souris. Cependant, ce n'est pas la méthode recommandée pour traiter les événements dans pygame, car il y a quelques défauts:

- Vous recevrez les états lorsque la fonction est appelée, ce qui signifie que vous pouvez manquer des événements entre les appels si l'utilisateur appuie sur les boutons rapidement.
- Vous ne pouvez pas déterminer l'ordre des événements.
- Vous devez toujours appeler l'une des fonctions d'événement de pygame pour que pygame puisse interagir en interne avec le système d'exploitation, sinon il vous avertira que le programme ne répond plus. Les fonctions que vous pouvez appeler sont:
    - `pygame.event.get()` pour obtenir tous les événements ou types d'événement (en transmettant les types en tant qu'argument) à partir de la file d'attente.
    - `pygame.event.poll()`

pour obtenir un seul événement de la file d'attente.
- `pygame.event.wait()` pour attendre un seul événement de la file d'attente.
- `pygame.event.clear()` pour effacer tous les événements de la file d'attente.
- `pygame.event.pump()` pour permettre à pygame de gérer les actions internes (est appelé implicitement par les fonctions ci-dessus).

Événements clavier

Le module clé a une fonction `pygame.key.get_pressed()` qui renvoie une liste de l'état de toutes les clés. La liste contient `0` pour toutes les touches qui ne sont pas enfoncées et `1` pour toutes les touches sur lesquelles vous appuyez. Son index dans la liste est défini par des constantes dans le module pygame, toutes préfixées par `K_` et le nom de la clé.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_a]:
    print("You pressed 'a'")
if key[pygame.K_F1]:
    print("You pressed 'F1'")
if key[pygame.K_LSHIFT]:
    print("You pressed 'left shift'")
if key[pygame.K_q]:  # Press 'q' to exit the program
    quit()
```

Si vous souhaitez rechercher une seule touche au lieu de la maintenir enfoncée, vous pouvez enregistrer l'état précédent de toutes les clés dans une variable temporaire et vérifier si la valeur change:

```
pygame.event.pump()  # Allow pygame to handle internal actions.
key = pygame.key.get_pressed()
if key[pygame.K_q] and not previous_key[pygame.K_q]:
    print("You pressed 'q'")
if key[pygame.K_p] and not previous_key[pygame.K_p]:
    print("You pressed 'p'")
previous_key = key
```

L'instruction ne vaut que lorsque la touche actuelle est enfoncée et que la touche précédente n'est pas enfoncée. Pour vérifier si l'utilisateur a relâché la clé, il suffit de changer le mot-clé `not` (`if not key[pygame.K_q] and previous_key[pygame.K_q]`). Pour que cela fonctionne correctement, vous devez définir la variable `previous_key = pygame.key.get_pressed()` avant la boucle du jeu, sinon vous recevrez une `NameError` .

Événements souris

Le module souris a des fonctions qui nous permettent de vérifier et de définir la position de la souris et de vérifier les boutons pressés. La fonction `pygame.mouse.get_pressed()` renvoie un tuple représentant si les boutons de la souris (gauche, molette de la souris, droite) sont enfoncés ou

non.

```
pygame.event.pump()  # Allow pygame to handle internal actions.
mouse_pos = pygame.mouse.get_pos()
mouse_buttons = pygame.mouse.get_pressed()
if mouse_pos[0] > 100:
    pygame.mouse.set_pos(10, mouse_pos[1])  # Reset the mouse's x-position to 10.
    print("YOU SHALL NOT PASS!")
if mouse_buttons[2]:
    print("I'm right, right?")
if mouse_buttons[0]:  # Press left mouse button to exit.
    print("Program left")
    quit()
```

Lire Gestion des événements en ligne: https://riptutorial.com/fr/pygame/topic/5110/gestion-des-evenements

Chapitre 9: Les essentiels

Examples

Dessin et animation de base

Ce programme dessine des formes et ' *bonjour le monde!* 'et laisser une image aller à chaque coin de la fenêtre.

le code complet:

```
import pygame,sys
from pygame.locals import *

pygame.init()

FPS = 30 #frames per second setting
fpsClock = pygame.time.Clock()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')

#set up the colors
black = (  0,   0,   0)
white = (255, 255, 255)
red   = (255,   0,   0)
green = (  0, 255,   0)
blue  = (  0,   0, 255)

imageImg  = pygame.image.load('baddie.png')
imagex = 320
imagey = 220
direction = 'left'

fontObj = pygame.font.Font('freesansbold.ttf', 32)
text = fontObj.render('Hello World!', True, green, blue)
rect = text.get_rect()
rect.center = (200, 150)

# the main game loop
while True:
    screen.fill(white)

    # draw a green polygon onto the surface
    pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0,
106)))

    # draw some blue lines onto the surface
    pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
    pygame.draw.line(screen, blue, (120, 60), (60, 120))
    pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)

    # draw a blue circle onto the surface
```

```
    pygame.draw.circle(screen, blue, (300, 50), 100, 0)

    # draw a red ellipse onto the surface
    pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)

    # draw a red rectangle onto the surface
    pygame.draw.rect(screen,red, (200, 150, 100, 50))

    # draw the text onto the surface
    screen.blit(text, rect)

    if direction == 'right':
        imagex += 5
        if imagex == 320:
            direction = 'down'
    elif direction == 'down':
        imagey += 5
        if imagey == 220:
            direction = 'left'
    elif direction == 'left':
        imagex -= 5
        if imagex == 20:
           direction = 'up'
    elif direction == 'up':
        imagey -= 5
        if imagey == 20:
           direction = 'right'
    screen.blit(imageImg, (imagex, imagey))

for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()

pygame.display.update()
fpsClock.tick(FPS)
```

mettre en place pygame et la fenêtre:

```
import pygame,sys
from pygame.locals import *

pygame.init()

#set up the window
screen = pygame.display.set_mode((500,400), 0, 32)
pygame.display.set_caption('drawing')
```

dessiner le fond blanc:

Dans cette fonction, vous définissez la couleur de l'arrière-plan.

```
screen.fill(white)
```

dessiner le polygone vert:

Vous définissez ici la surface d'affichage, la couleur et la position de chaque coin du polygone (coordonnées x et y), que vous pouvez faire dans le sens des aiguilles d'une montre et dans le sens inverse.

```
pygame.draw.polygon(screen, green, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))
```

dessiner les lignes bleues:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, le premier et le dernier point et la largeur de la ligne (si vous ne donnez pas de largeur, c'est juste 1).

```
pygame.draw.line(screen, blue, (60, 60), (120, 60), 4)
pygame.draw.line(screen, blue, (120, 60), (60, 120))
pygame.draw.line(screen, blue, (60, 120), (120, 120), 4)
```

dessiner le cercle bleu:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position, le rayon et la largeur du cercle (si vous donnez un 0 pour la largeur, c'est un cercle simple).

```
pygame.draw.circle(screen, blue, (300, 50), 100, 0)
```

dessiner l'ellipse:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position, la taille horizontale et la taille et la largeur verticales.

```
pygame.draw.ellipse(screen, red, (300, 250, 80,80), 1)
```

dessiner le rectangle:

Dans cette fonction, vous définissez la surface d'affichage, la couleur, la position et la taille horizontale et verticale.

```
pygame.draw.rect(screen,red, (200, 150, 100, 50))
```

définir le texte:

Vous définissez d'abord le type et la taille de votre texte avec cette fonction:

```
fontObj = pygame.font.Font('freesansbold.ttf', 32)
```

Ensuite, vous définissez le texte réel, si le texte est en gras, la couleur et, si vous le souhaitez, la couleur du marquage. Vous pouvez le faire avec cette fonction:

```
text = fontObj.render('Hello World!', True, green, blue)
```

Si vous souhaitez marquer votre texte, vous devez dire à pygame que cette fonction:

```
rect = text.get_rect()
```

Et si vous voulez définir la position du centre du texte, vous pouvez le faire avec cette fonction:

```
rect.center = (200, 150)
```

dessiner le texte:

Si vous avez défini un marquage et / ou le centre:

```
screen.blit(text, rect)
```

Sinon, vous devez définir la position du texte afin de dessiner le texte de cette façon:

```
screen.blit(text, (100,50))
```

définir l'image:

Vous définissez ici l'image que vous souhaitez utiliser (si vous le faites de cette façon, le fichier image doit se trouver dans le même répertoire que votre fichier programme), la position de départ (x et y) et la direction de l'image.

```
image  = pygame.image.load('image.png')
baddiex = 320
baddiey = 220
direction = 'left'
```

animer l'image:

Avec cette partie du code, nous vérifions la direction de l'image, si elle a atteint un coin, si c'est le cas, changez de direction, sinon tracez l'image 5 pixels plus loin dans la même direction.

```
if direction == 'right':
    imagex += 5
    if imagex == 360:
        direction = 'down'
elif direction == 'down':
    imagey += 5
    if imagey == 260:
        direction = 'left'
elif direction == 'left':
    imagex -= 5
    if imagex == 20:
       direction = 'up'
elif direction == 'up':
    imagey -= 5
    if imagey == 20:
       direction = 'right'
screen.blit(imageImg, (imagex, imagey))
```

note: Mon image est 20x20 pixels, j'utilise `if imagex == 360:` et `if imagey == 260:` car alors mon image est à 20 pixels du bord de la fenêtre, tout comme les 2 autres coins. Si votre image a une taille différente, vous devrez probablement modifier ces chiffres.

vérification de quit:

Ici, nous vérifions si vous avez fermé la fenêtre Pygame, et si oui, fermez la fenêtre, si vous n'écrivez pas cela quelque part dans votre programme, vous ne pourrez probablement pas fermer la fenêtre.

```
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()
```

mettre à jour l'écran:

Avec cette fonction, vous mettez à jour l'écran pour que tout ce que vous avez dessiné devienne visible.

```
pygame.display.update()
```

Réglage FPS:

Avec cette fonction, vous dites à Pygame de dormir suffisamment pour que votre réglage FPS soit respecté.

```
fpsClock.tick(FPS)
```

Utilisation avec PIL

Lorsque vous devez utiliser à la fois PIL et Pygame, car il manque des fonctionnalités dans les deux, vous avez besoin d'un moyen de convertir entre les surfaces Pygame et les images PIL, de préférence sans les écrire sur le disque.

Pour cela, vous pouvez utiliser les fonctions "tostring" et "fromstring" fournies dans les deux bibliothèques.

Conversion de PIL à Pygame:

```
strFormat = 'RGBA'
raw_str = image.tostring("raw", strFormat)
surface = pygame.image.fromstring(raw_str, image.size, strFormat)
```

Conversion de Pygame à PIL:

```
strFormat = 'RGBA'
raw_str = pygame.image.tostring(surface, strFormat, False)
image = Image.frombytes(strFormat, surface.get_size(), raw_str)
```

Lire Les essentiels en ligne: https://riptutorial.com/fr/pygame/topic/4196/les-essentiels

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec pygame	Community, elegent, Inazuma, Nearoo, Ni., numbermaniac, svs, Ted Klein Bergman, White Shadow
2	Ajout de musique de fond et d'effets sonores	Hikaryu, White Shadow
3	Créer une fenêtre dans pygame - pygame.display.set_mode ()	Nearoo
4	Créer une fenêtre Pygame	Rishi Malhotra, White Shadow
5	Créer une fenêtre simple de pygame	ModoUnreal
6	Dessin sur l'écran	svs
7	Gestion des événements	elegent, Mikhail V, Nearoo, Ted Klein Bergman
8	Les essentiels	Ni., numbermaniac, svs, Ted Klein Bergman