EBook Gratuito

APPRENDIMENTO pygame

Free unaffiliated eBook created from
Stack Overflow contributors.

#pygame

Sommario

Di

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

Capitolo 1: Iniziare con pygame

Osservazioni

Pygame è un wrapper Python per SDL - una libreria C multipiattaforma per il controllo multimediale -, scritto da Pete Shinners. Ciò significa che, usando pygame, puoi scrivere videogiochi o altre applicazioni multimediali in Python che verranno eseguite inalterate su qualsiasi piattaforma supportata da SDL (Windows, Unix, Mac, BeOS e altri).

Questa sezione fornisce una panoramica su cosa sia Pygame e perché uno sviluppatore potrebbe volerlo usare.

Dovrebbe anche menzionare qualsiasi argomento di grandi dimensioni all'interno di pygame e collegarsi agli argomenti correlati. Poiché la documentazione di pygame è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	==========>	Data di rilascio
Pygame 1.9.0	========>	1 agosto 2009
Pygame 1.8.1	========>	30 luglio 2008
Pygame 1.8.0	========>	29 marzo 2008
Pygame 1.7.1	========>	16 agosto 2005
Pygame 1.6.2	========>	-
Pygame 1.6	=========>	23 ottobre 2003
Pygame 1.5	=========>	30 maggio 2002
Pygame 1.4	=========>	30 gen 2002
Pygame 1.3	=========>	dic 19, 2001
Pygame 1.2	=========>	Set 4, 2001
Pygame 1.1	=========>	23 giugno 2001
Pygame 1.0	=========>	5 aprile 2001
Pygame 0.9	=========>	Feb 13, 2001
Pygame 0.5	=========>	Gen 6 14, 2001
Pygame 0.4	=========>	Dec 14, 2000
Pygame 0.3	=========>	Nov 20, 2000
Pygame 0.2	=========>	Nov 3, 2000
Pygame 0.1	=========>	28 ottobre 2000

Examples

Un semplice 'gioco'

Importa e inizializza

Ogni modulo deve essere importato e pygame non fa eccezione. Sebbene sia necessario inizializzare correttamente la funzione `pygame.init()` per tutti i moduli importati in pygame. Se dimentichiamo che alcuni moduli non funzioneranno. La funzione restituisce anche una tupla di tutte le inizializzazioni riuscite e fallite (non genera un errore se un modulo non riesce a inizializzarsi).

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

Crea necessità

Abbiamo anche bisogno di creare un display. Pygame ha già creato un display (nascosto), quindi tutto ciò che dobbiamo fare è impostare la modalità del display (in questo esempio impostiamo solo la risoluzione). È anche una buona idea creare un orologio per assicurarci che il nostro programma si aggiorni a una velocità fissa (altrimenti funzionerebbe a velocità diverse a seconda della velocità del computer).

```
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.
```

Per un po 'di leggibilità più avanti nel nostro codice creeremo due costanti di colore, che rappresentano una tupla di rosso, verde e blu (RGB). I valori vanno da 0 (nessuna luce) a 255 (luce piena).

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

In pygame usiamo solitamente una *superficie* per rappresentare l'aspetto di un oggetto e un *rettangolo* (*Rect*) per rappresentare la posizione di un oggetto. Una *superficie* è come un foglio di carta bianco che contiene colori o immagini. Se stai facendo una lezione dovresti nominare gli attributi *image* e *rect* poiché molte funzioni cercheranno e useranno quegli attributi. Tali classi trarrebbero vantaggio ereditando la classe `pygame.sprite.Sprite` per motivi che puoi leggere qui .

```
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).
```

Il ciclo di gioco

Ora abbiamo tutto pronto per il nostro ciclo di gioco. Questo è un ciclo che verrà eseguito per l'intero gioco, dove gestiamo eventi e aggiorniamo lo schermo e le posizioni dei nostri oggetti.

Per prima cosa ci assicureremo che il nostro ciclo venga eseguito a un determinato *FPS* . Abbiamo definito l' *FPS* e creato il nostro orologio all'inizio del programma. Il seguente codice farà in modo che il nostro programma abbia abbastanza tempo per fare in modo che il nostro loop ripeta la quantità che abbiamo definito il nostro *FPS* . In questo esempio, 60 volte al secondo.

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

Quindi gestiremo eventi. Un evento è fondamentalmente un'azione dell'utente, come lo spostamento del mouse o la pressione di un tasto. Pygame registrerà tutti questi eventi in una coda che otteniamo chiamando `pygame.event.get()` . Possiamo iterare su questo e controllare se c'è un evento che vorremmo gestire. Gli eventi hanno un attributo *type* che possiamo controllare rispetto alle costanti nel modulo pygame per determinare quale tipo di evento sia.

```
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.
```

Possiamo anche controllare `if event.type == pygame.KEYDOWN` per vedere se l'utente ha premuto un tasto. In tal caso l'evento ha una *chiave di* attributo che possiamo controllare per vedere quale chiave rappresenta.

```
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)
```

Ora dobbiamo mostrare la nostra immagine. Per prima cosa potremmo voler cancellare il nostro schermo dal rendering precedente. Facciamo così riempiendo il nostro intero schermo con il nero (rimuovi il codice per capire perché vogliamo eliminarlo). Quindi dobbiamo *confondere la* nostra *immagine* sullo schermo. Blittare significa essenzialmente copiare l' *immagine* su un'altra superficie (nel nostro caso, lo schermo). Infine si *flip* o *aggiornare* lo schermo.

Quando stiamo blittando, in realtà non stiamo visualizzando nulla all'utente. Immaginalo come il computer su un lato e l'utente sull'altro. Il computer disegna (*blits*) sul suo lato dello schermo, lo *gira* verso l'utente e quindi si ripete.

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

Ora abbiamo un gioco di base! Abbastanza noioso, sì, ma gli elementi essenziali sono lì! Combinato con la tua attuale conoscenza di Python e dovresti essere in grado di creare qualcosa di fantastico.

Codice completo

```
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()
```

Meccaniche di gioco leggermente migliorate

Si noti che il programma verifica quando si preme il tasto e non quando si tiene premuto il tasto. Per risolvere questo problema potremmo introdurre una variabile di *velocità* . Possiamo creare una classe di giocatori per mantenerla più organizzata. Per evitare il movimento dipendente dalla trama (se modifichiamo l'FPS a 30 gli oggetti si sposterebbero a metà della velocità) introduciamo il movimento dipendente dal tempo passando il tempo tra le zecche ai nostri oggetti mobili.

```
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.
```

Ci sono ancora molte cose che dovrebbero essere migliorate su questo codice. Ti consiglio di leggere il tutorial su pygame e questo discorso di Richard Jones per ulteriori approfondimenti.

Installare pygame

Su Windows

1. Passare a http://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame - un sito non ufficiale che fornisce binari di Windows di pacchetti Python open source per la distribuzione CPython ufficiale di *Christoph Gohlke* .

2. Scarica il file `.whl` pygame appropriato in base alla tua versione python installata. (Il file è chiamato qualcosa come `pygame - <pygame version> - <python version> - win32.whl`)

3. Correre

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

   nel tuo terminale, bash o consol.
   Nota: se `pip` non viene trovato in `PATH` prova ad eseguire `python -m pip install your-pygame-package.whl`

4. Controlla se puoi importare pygame come un modulo python

   ```
   import pygame
   ```

   Se non ricevi un errore, hai correttamente installato pygame sul tuo computer :)

Su linux

1. Apri il tuo terminale ed esegui

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

   Nota: questo installerà pygame per python2

2. Prova ad importare pygame all'interno

   ```
   import pygame
   ```

   Se non ricevi un errore, hai correttamente installato pygame sul tuo sistema linux :)

Su macOS

Ci sono due modi per installarlo su mac:

Metodo 1

Vai alla pagina dei download di Pygame e scarica il programma di installazione mac. Eseguilo e dovrebbe installare Pygame sul tuo Mac.

Metodo 2

Installa Homebrew :

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

Quindi usa Homebrew per installare Python 2.7.12 e Pygame:

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

Ora esegui Python nel tuo terminale e prova a `import pygame` . Se non dice nulla, è installato con successo.

Importazione di pygame e disegno su un display

Iniziare

Devi fare quanto segue per iniziare con Pygame:

```
import pygame
```

Questo apre una finestra di dimensioni 640.480 e lo memorizza in una variabile chiamata schermata.

Impostazione del nome di una finestra

L'impostazione di un nome per la finestra pygame richiede la seguente sintassi:

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

A proposito dello schermo

- Il punto (0,0) si trova nell'angolo in alto a sinistra dello schermo.
- le coordinate x aumentano da sinistra a destra, le coordinate y aumentano dall'alto verso il basso. Quelle coordinate sul lato destro sul piano cartesiano sono positive e il lato sinistro è negativo. Tuttavia, le coordinate del lato superiore sul piano cartesiano sono negative sopra e positive in basso. (**Nota** : questo è considerato se i punti sono presi dall'origine.)

Aggiornamento dello schermo

Le modifiche apportate allo schermo, ad esempio riempiendole di colore o disegnandole, non vengono visualizzate immediatamente!
Quindi come si fa?

Devi chiamare questa funzione:

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

Colori

La colorazione in pygame funziona in modalità RGB.
Il codice per la colorazione è:

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

- R sta per rosso.
- G sta per verde
- B sta per blu.
- Tutti e tre dovrebbero essere numeri interi compresi tra 0 e 255, con 255 più luminosi e 0 più scuro

Disegno

1. Per disegnare linee

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

2. Per disegnare il rettangolo

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

3. Per disegnare il cerchio

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

Impostare tutto in un ciclo

Per fare un ciclo usa il seguente codice:

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

Disegnare un rettangolo sulla finestra di

pygame (codice)

```
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()
```

Leggi Iniziare con pygame online: https://riptutorial.com/it/pygame/topic/3959/iniziare-con-pygame

Capitolo 2: Aggiunta di musica di sottofondo e effetti sonori

Osservazioni

Prova a riprodurre musica in ".wav" anziché ".mp3". In ".mp3" la musica è in ritardo.

Examples

Esempio per aggiungere musica in 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.
```

Esempio per aggiungere playlist musicali in 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
```

Leggi Aggiunta di musica di sottofondo e effetti sonori online:
https://riptutorial.com/it/pygame/topic/7419/aggiunta-di-musica-di-sottofondo-e-effetti-sonori

Capitolo 3: Creare una finestra in pygame - pygame.display.set_mode ()

Sintassi

- pygame.display.set_mode (resolution = (0,0), flags = 0, depth = 0) # Restituisce un pygame.Surface che rappresenta la finestra sullo schermo
- flags = pygame.FULLSCREEN | pygame.OPENGL # Le bandiere possono essere combinate usando "|" (carattere OR bit o "pipe").

Parametri

parametro	spiegazione
risoluzione	una coppia di numeri che rappresentano la larghezza e l'altezza della finestra
bandiere	opzioni aggiuntive che cambiano il tipo di finestra - vedi "Note" per le bandiere disponibili
profondità	quantità di bit utilizzati per il colore

Osservazioni

- I possibili valori per gli argomenti `flag` sono:

bandiera	descrizione
pygame.FULLSCREEN	la finestra è a schermo intero
pygame.RESIZABLE	la finestra è ridimensionabile
pygame.NOFRAME	la finestra non ha bordi o controlli
pygame.DOUBLEBUF	usa il doppio buffer - raccomandato per `HWSURFACE` o `OPENGL`
pygame.HWSURFACE	finestra è accelerata hardware, possibile solo in combinazione con `FULLSCREEN`
pygame.OPENGL	la finestra è renderizzabile da OpenGL

Altre osservazioni:

- Attualmente Pygame può gestire solo una singola finestra alla volta. La creazione di una seconda finestra chiamando `pygame.display.set_mode((x,y))` una seconda volta chiuderà la

prima finestra.

- La modifica dell'argomento delle `depths` non è quasi mai richiesta: pygame selezionerà la migliore da sola. Nel caso in cui sia impostata una profondità non supportata dal sistema, pygame emulerà questa profondità, che può essere molto lenta.
- Le cose che sono disegnate sulla superficie restituite da `pygame.display.set_mode()` non sono immediatamente visibili sullo schermo - il display deve prima essere capovolto usando `pygame.display.update()` o `pygame.display.flip()` .

Examples

Crea una finestra Pygame

Questo crea una finestra a schermo intero con dimensioni 500x500 pixel:

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

`screen` rappresenta da ora in poi la finestra sullo schermo; è un oggetto pygame.Surface. Tutto ciò che dovrebbe essere visibile all'utente deve essere disegnato su di esso usando `screen.blit` .

Leggi Creare una finestra in pygame - pygame.display.set_mode () online: https://riptutorial.com/it/pygame/topic/6442/creare-una-finestra-in-pygame---pygame-display-set-mode---

Capitolo 4: Creare una semplice finestra pygame

Examples

Il codice completo

```
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()
```

Importazione e inizializzazione di pygame

Come facciamo con qualsiasi modulo in python, dobbiamo importare pygame:

```
import pygame
```

Inizializziamo quindi tutti i moduli pygame importati:

```
pygame.init()
```

Questo è usato per inizializzare tutti i moduli pygame. Senza questo i moduli non funzionerebbero

Definire costanti

Quindi definiamo alcune costanti qui:

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

Le costanti `WIDTH` e `HEIGHT` vengono utilizzate per creare una finestra, che avrebbe una larghezza di 300 pixel e un'altezza di 200 pixel. La funzione utilizzata in `SCREEN` , `pygame.display.set_mode((WIDTH, HEIGHT))` , imposta la modalità del display e restituisce un oggetto Surface . Nota come i parametri per questa funzione sono le costanti `WIDTH` e `HEIGHT` definite in precedenza.

Impostazione del nome della finestra

Usiamo quindi questa funzione per cambiare il nome della finestra in Il mio gioco:

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

Definizione dei colori

Successivamente definiamo 6 colori che possono essere utilizzati nella nostra finestra:

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

Quando definiamo i colori, inseriamo 3 valori compresi tra 0 e 255. La classe pygame.Color normalmente va in questo formato:

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

Dove il parametro r imposta il valore rosso del colore, il parametro g imposta il valore verde del colore e il parametro b imposta il valore blu del colore. Il parametro a imposta il valore alfa del colore.

Diamo quindi questo comando:

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

Questa è una funzione pygame.Surface.fill che riempie l'oggetto Surface, il nostro schermo, con il colore rosso.

Utilizzando pygame.display.flip ()

Quindi usiamo questa funzione

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

In pratica, tutto ciò che abbiamo disegnato sullo schermo diventa visibile e aggiorna il contenuto dell'intero display. Senza questa linea, l'utente non vedrebbe nulla sulla loro schermata pygame.

Il ciclo di gioco

Le righe successive sono quelle che vengono definite "loop di gioco".

Per iniziare, creiamo una variabile e rendiamo True:

```
is_running = True
```

In modo che possiamo iniziare il nostro ciclo while:

```
while is_running:
```

che sarà in esecuzione per tutto il gioco.

Nella sua forma più elementare, pygame ha "eventi" che prendono l'input dell'utente, ad esempio una pressione di un pulsante o un clic del mouse. Pygame gestisce questi eventi attraverso una coda di eventi. Possiamo ottenere questi eventi dalla coda degli eventi con questo ciclo for:

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

Che fondamentalmente passa attraverso un elenco di eventi, la nostra coda di eventi. Queste sono le prossime 2 righe:

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

Questo farà in modo che quando l'utente preme il pulsante di uscita nell'angolo superiore, si verifichi l'evento con il tipo `pygame.QUIT` .

Questo termina quindi il ciclo while, dato che `is_running` ora è `False` e lo script passa alla riga finale:

```
pygame.quit()
```

Quale non inizializza i moduli pygame.

Leggi Creare una semplice finestra pygame online:
https://riptutorial.com/it/pygame/topic/6597/creare-una-semplice-finestra-pygame

Capitolo 5: Creazione di una finestra Pygame

Osservazioni

Se si desidera avere altri colori come sfondo, quindi denominare una nuova variabile come `red = (255,0,0)` e modificare `display.fill(black)` per `display.fill(red)` . È possibile creare colori memorizzandoli in una variabile e controllando i loro valori RGB da Internet.

Examples

Creazione della finestra 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()
```

Leggi Creazione di una finestra Pygame online:
https://riptutorial.com/it/pygame/topic/6477/creazione-di-una-finestra-pygame

Capitolo 6: Disegnare sullo schermo

Examples

disegnare forme, testo e immagini sullo schermo con una piccola animazione

Questo programma disegnerà alcune forme sul display, disegnerà "Ciao mondo!" nel mezzo dello schermo e lascia che un'immagine arrivi ad ogni angolo della finestra. È possibile utilizzare ogni immagine desiderata, ma **è necessario posizionare il file immagine nella stessa directory del programma.**

l'intero codice:

```
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)
```

disegnando lo sfondo bianco:

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

disegnando il poligono:

In questa funzione definisci la superficie di visualizzazione, il colore e la posizione di ogni angolo del poligono, puoi farlo in senso orario e antiorario.

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

disegnando le linee:

Qui definisci la superficie di visualizzazione, il colore, il primo e l'ultimo punto e la larghezza della linea.

```
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)
```

disegnare il cerchio:

In questa funzione si definisce la superficie di visualizzazione, il colore, la posizione, il raggio e la larghezza del cerchio (0 dà un cerchio semplice).

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

disegnando l'ellisse:

In questa funzione definisci la superficie di visualizzazione, il colore, la posizione, la dimensione orizzontale, la dimensione verticale e la larghezza dell'ellisse

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

disegnando il rettangolo:

In questa funzione si definisce la superficie di visualizzazione, il colore, la posizione e la dimensione verticale e orizzontale del rettangolo.

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

definizione del testo:

Per prima cosa definisci il tipo e la dimensione del tuo testo, io uso un font di base che ottieni con pygame.

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

Quindi definisci il testo attuale, se lo vuoi in grassetto o no (Vero / Falso), il colore del testo e, se vuoi segnare il tuo testo, un colore del segno.

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

Se vuoi segnare il tuo testo o vuoi definire il centro del tuo testo, devi dire a pygame che con questa funzione:

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

E dopo, puoi definire il centro del tuo testo con questa funzione:

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

disegno del testo:

Se hai segnato il tuo testo o definito il centro, devi disegnare il testo in questo modo:

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

Altrimenti si disegna il testo, ma è necessario definire la posizione, in modo da farlo in questo modo:

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

definizione dell'immagine:

Qui definisci l'immagine che vuoi usare, la posizione iniziale (coordinate xey) e la direzione dell'immagine.

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

animare l'immagine:

Qui si controlla la direzione dell'immagine, se ha raggiunto un angolo, in tal caso cambia la direzione, in caso contrario spostalo di 5 pixel nella stessa direzione e disegna nuovamente l'immagine. Questo è ciò che facciamo con questa parte del codice:

```
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))
```

nota: la mia immagine è 20 per 20 pixel, l'ho usata `if imagex == 360` e `if imagey == 260:` perché allora la mia immagine è a 20 pixel dal bordo, se l'immagine ha una dimensione diversa, dovrai cambiare i numeri .

controllando se si esce dal programma:

Qui controlliamo se hai chiuso la finestra del tuo programma.

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

aggiornamento del display:

Qui dici a pygame di aggiornare il display in modo che tutto ciò che hai disegnato appaia sullo schermo.

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

definendo i frame al secondo:

Qui dici a pygame di dormire abbastanza in modo da rispettare l'impostazione dei fotogrammi al secondo.

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

Leggi Disegnare sullo schermo online: https://riptutorial.com/it/pygame/topic/6287/disegnare-sullo-schermo

Capitolo 7: Disegnare sullo schermo

Sintassi

- pygame.draw.rect (Superficie, colore, Rettangolo, larghezza = 0)
- pygame.draw.polygon (Superficie, colore, lista puntiforme, larghezza = 0)
- pygame.draw.circle (Superficie, colore, pos, raggio, larghezza = 0)
- pygame.draw.ellipse (Superficie, colore, Rettangolo, larghezza = 0)
- pygame.draw.arc (Superficie, colore, Rect, start_angle, stop_angle, width = 1)
- pygame.draw.line (Superficie, colore, start_pos, end_pos, larghezza = 1)
- pygame.draw.lines (Superficie, colore, chiuso, lista puntiforme, larghezza = 1)
- pygame.draw.aaline (Superficie, colore, startpos, endpos, blend = 1)
- pygame.draw.aalines (Superficie, colore, chiuso, elenco punti, blend = 1)

Parametri

parametri	Dettagli
Superficie	La superficie su cui disegnare la forma.
colore	Una sequenza di 3 o 4 interi che rappresenta il rosso, il verde e il blu (e l'alfa), ciascun valore compreso tra 0-255.
Rect	Un'area rettangolare in cui verrà disegnata la forma.
larghezza	La larghezza delle linee. La forma sarà riempita se larghezza = 0.
pointlist	Un elenco di una quantità arbitraria di punti / vertici, in pixel (x, y).
pos	La posizione del centro del cerchio, in pixel (x, y).
raggio	Il raggio dei cerchi in pixel.
chiuso	Se è vero, verrà disegnata una linea tra l'ultimo e il primo punto, chiudendo la forma.
fondere = 1	Se è vero, le sfumature verranno sfumate con le sfumature di pixel esistenti invece di sovrascriverle.
start_angle	L'angolo iniziale dell'arco, in radianti.
stop_angle	L'angolo finale dell'arco, in radianti.
start_pos	La posizione iniziale della linea, in pixel.
end_pos	La posizione finale della linea, in pixel

Examples

Disegnare con il modulo di disegno

Pygame ha un modulo, `pygame.draw` , che contiene funzioni che possono disegnare forme direttamente su una superficie.

Funzione	Descrizione
pygame.draw.rect	disegna una forma rettangolare
pygame.draw.polygon	disegnare una forma con qualsiasi numero di lati
pygame.draw.circle	Disegna un cerchio attorno a un punto
pygame.draw.ellipse	disegna una forma rotonda all'interno di un rettangolo
pygame.draw.arc	disegna una sezione parziale di un'ellisse
pygame.draw.line	disegnare un segmento di linea retta
pygame.draw.lines	disegnare più segmenti di linea contigui
pygame.draw.aaline	disegnare sottili linee antialias
pygame.draw.aalines	disegnare una sequenza collegata di linee antialias

Come usare il modulo

Per usare il modulo devi prima importare e inizializzare correttamente pygame e impostare una modalità per la visualizzazione. È comodo definire le costanti del colore in anticipo, rendendo il codice più leggibile e più bello. Tutte le funzioni impiegano una superficie per disegnare, un colore e un argomento di posizione che è sia un pygame Rect o una sequenza di numeri interi / float di 2 elementi (il `pygame.draw.circle` prenderà solo interi a causa di motivi indefiniti).

Esempio

Il codice seguente mostrerà tutte le diverse funzioni, come vengono utilizzati e come appaiono. Inizializzeremo pygame e definiremo alcune costanti prima degli esempi.

```
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)
```

Il colore nero è il colore di default della superficie e rappresenta la parte della superficie su cui non è stata disegnata. I parametri di ciascuna funzione sono spiegati di seguito a **Parametri** .

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())
```

Poligono

```
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)
```

Cerchio

```
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)
```

I buchi sono una sfortunata conseguenza dell'algoritmo di disegno di Pygame.

Ellisse

```
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())
```

I buchi sono una sfortunata conseguenza dell'algoritmo di disegno di Pygame.

Arco

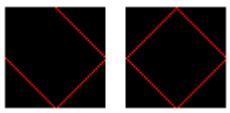
```
size = (50, 50)

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

Linea

```
size = (50, 50)

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

Linee

```
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)
```

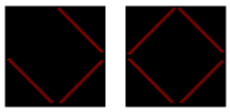


Linea antialias

```
size = (50, 50)

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

Linee antialias

```
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)
```



Provalo

Per provarlo tu stesso: copia uno dei frammenti di codice sopra e il codice sottostante in un file vuoto, cambia l' *immagine* del nome con il nome della superficie che vuoi miscelare e sperimentare.

```
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()
```

superfici

In pygame di solito usi le Superfici per rappresentare l'aspetto degli oggetti, e Rettangoli per rappresentare le loro posizioni. Una superficie è come un foglio di carta bianco che contiene colori o immagini. Esistono due modi per creare una superficie: vuota da zero o caricando un'immagine.

Crea una superficie

Per creare una superficie è necessario almeno la sua dimensione, che è una sequenza intera di 2 elementi di larghezza e altezza, che rappresenta la dimensione in pixel.

È inoltre possibile passare argomenti aggiuntivi durante la creazione di una superficie per controllare profondità di bit, maschere e funzioni aggiuntive come alfa per pixel e / o creare l'immagine nella memoria video. Questo è al di fuori dello scopo di questo esempio però.

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

È possibile utilizzare il modulo `pygame.draw` per disegnare forme sulla superficie o riempirlo con un colore chiamando il `fill(color)` metodo Surface `fill(color)` . Il *colore* dell'argomento è una sequenza di numeri interi a 3 o 4 elementi o un oggetto `pygame.Color` .

Carica un'immagine

Molto spesso ti piacerebbe utilizzare le tue immagini in un gioco (chiamato sprites). Creare una superficie con la tua immagine è facile come:

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

Il percorso dell'immagine può essere relativo o assoluto. Per migliorare le prestazioni, in genere è consigliabile convertire l'immagine nello stesso formato di pixel dello schermo. Questo può essere fatto chiamando il metodo Surface `convert()` , in questo modo:

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

Se l'immagine contiene trasparenza (valori alfa), si chiama invece il metodo `convert_alpha()` :

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

blitting

Le superfici devono essere masticate allo schermo per poterle visualizzare. Blittare significa essenzialmente copiare i pixel da una superficie all'altra (anche lo schermo è una superficie). È inoltre necessario passare la posizione di Surface, che dovrebbe essere una sequenza di numeri interi a 2 elementi o un oggetto Rect. Il topleft di Surface verrà posizionato nella posizione.

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

È possibile eseguire il blit su altre Superfici rispetto allo schermo. Per visualizzare ciò che è stato visualizzato sullo schermo, devi chiamare `pygame.display.update()` o `pygame.display.flip()` .

Trasparenza

Sono supportati i tipi 3 di trasparenza in pygame: colorkeys, alfa di Surface e alpha per pixel.

Colorkeys

Rende un colore completamente trasparente o, più esattamente, rendendo semplicemente un colore non facile. Se hai un'immagine con un rect nero all'interno puoi impostare un colorkey per evitare che il colore nero sia blitante.

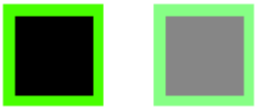
```
BLACK = (0, 0, 0)
my_image.set_colorkey(BLACK)  # Black colors will not be blit.
```

Una superficie può avere solo una colorkey. L'impostazione di un'altra colorkey sovrascriverà la precedente. I colorkeys non possono avere valori alfa diversi, possono solo rendere un colore non visibile.

Alfa di superficie

Rende trasparente l'intera superficie con un valore alfa. Con questo metodo è possibile avere valori alfa diversi ma interesserà l'intera superficie.

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



Per pixel alfa

Rende trasparenti tutti i pixel della superficie con un valore alfa individuale. Questo ti dà la massima libertà e flessibilità, ma è anche il metodo più lento. Questo metodo richiede anche che Surface sia creato come una superficie alfa per pixel e che gli argomenti colore debbano contenere un quarto intero alfa.

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

La superficie ora disegnerà trasparenza se il colore contiene il quarto valore alfa.

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

A differenza delle altre Superfici, questo colore di default di Surface non sarà nero ma trasparente. Ecco perché il rettangolo nero nel mezzo scompare.

Combina colorkey e Surface alpha

Colorkeys e alfa di superficie possono essere combinati, ma non possono essere eseguiti per pixel alfa. Questo può essere utile se non vuoi le prestazioni più lente di una superficie per pixel.

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

Codice completo

Copia questo in un file vuoto ed eseguilo. Premere i tasti 1, 2, 3 o 4 per visualizzare le immagini. Premi 2, 3 o 4 più volte per renderli più opachi.

```
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()
```

Leggi Disegnare sullo schermo online: https://riptutorial.com/it/pygame/topic/7079/disegnare-sullo-schermo

Capitolo 8: L'essenziale

Examples

Disegno e animazione di base

Questo programma disegna alcune forme e ' *ciao mondo!* 'e lascia che un'immagine arrivi ad ogni angolo della finestra.

il codice completo:

```
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)
```

impostazione di pygame e della finestra:

```
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')
```

disegnando lo sfondo bianco:

In questa funzione si definisce il colore dello sfondo.

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

disegnando il poligono verde:

Qui puoi definire la superficie di visualizzazione, il colore e la posizione di ogni angolo del poligono (coordinate xey), puoi farlo in senso orario e antiorario.

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

disegnando le linee blu:

In questa funzione si definisce la superficie di visualizzazione, il colore, il primo e l'ultimo punto e la larghezza della linea (se non si dà una larghezza, è solo 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)
```

disegnando il cerchio blu:

In questa funzione si definisce la superficie di visualizzazione, il colore, la posizione, il raggio e la larghezza del cerchio (se si assegna uno 0 per la larghezza, si tratta di un cerchio semplice).

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

disegnando l'ellisse:

In questa funzione si definisce la superficie di visualizzazione, il colore, la posizione, la dimensione orizzontale e la dimensione verticale e la larghezza.

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

disegnando il rettangolo:

In questa funzione si definisce la superficie di visualizzazione, il colore, la posizione e la dimensione orizzontale e verticale.

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

definizione del testo:

Per prima cosa definisci il tipo e la dimensione del tuo testo con questa funzione:

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

Quindi definisci il testo attuale, se il testo è in grassetto, il colore e, se vuoi, il colore del segno. Puoi farlo con questa funzione:

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

Se vuoi segnare il tuo testo, devi dire a pygame che con questa funzione:

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

E se vuoi definire la posizione del centro del testo puoi farlo con questa funzione:

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

disegno del testo:

Se hai definito una marcatura e / o il centro:

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

Altrimenti devi definire la posizione del testo quindi disegna il testo in questo modo:

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

definizione dell'immagine:

Qui definisci quale immagine vuoi usare (se lo fai in questo modo, il file immagine deve essere nella stessa directory del file di programma), la posizione iniziale (xey) e la direzione dell'immagine.

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

animare l'immagine:

Con questa parte del codice controlliamo la direzione dell'immagine, se raggiunge un angolo, in tal caso, cambia la direzione, altrimenti, disegna l'immagine di 5 pixel ulteriormente nella stessa direzione.

```
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))
```

nota: la mia immagine è di 20x20 pixel, io uso `if imagex == 360:` e `if imagey == 260:` perché allora la mia immagine è di 20 pixel dal bordo della finestra, proprio come gli altri 2 angoli. Se la tua immagine ha una dimensione diversa, probabilmente dovrai cambiare quei numeri.

controllare per uscire:

Qui controlliamo se hai chiuso la finestra di pygame e, in tal caso, chiudi la finestra, se non la scrivi da qualche parte nel tuo programma, probabilmente non sarai in grado di chiudere la finestra.

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

aggiornare lo schermo:

Con questa funzione si aggiorna lo schermo in modo che tutto ciò che si è disegnato sia visibile.

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

Impostazione FPS:

Con questa funzione devi dire a pygame di dormire abbastanza in modo che la tua impostazione FPS sia rispettata.

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

Usando con PIL

Quando devi usare sia PIL che Pygame perché mancano le funzionalità in entrambi, hai bisogno di un modo per convertire tra Pygame Surfaces e PIL Immagini, preferibilmente senza scriverle sul disco.

Per questo è possibile utilizzare le funzioni "tostring" e "fromstring" fornite in entrambe le librerie.

Conversione da PIL a Pygame:

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

Conversione da Pygame a PIL:

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

Leggi L'essenziale online: https://riptutorial.com/it/pygame/topic/4196/l-essenziale

Capitolo 9: Manipolazione degli eventi

Examples

Ciclo degli eventi

Pygame registrerà tutti gli eventi dell'utente in una coda di eventi che può essere ricevuta con il codice `pygame.event.get()` . Ogni elemento in questa coda è un oggetto `Event` e avranno tutti il `type` attributo, che è un numero intero che rappresenta il tipo di evento. Nel modulo pygame ci sono costanti integer predefinite che rappresentano il tipo. Tranne questo attributo, gli eventi hanno attributi diversi.

Nome costante	attributi
SMETTERE	nessuna
ACTIVEEVENT	guadagno, stato
keydown	unicode, chiave, mod
keyup	chiave, mod
MOUSEMOTION	pos, rel, buttons
MOUSEBUTTONUP	pos, pulsante
MOUSEBUTTONDOWN	pos, pulsante
JOYAXISMOTION	gioia, asse, valore
JOYBALLMOTION	gioia, palla, rel
JOYHATMOTION	gioia, cappello, valore
JOYBUTTONUP	gioia, pulsante
JOYBUTTONDOWN	gioia, pulsante
VIDEORESIZE	taglia, w, h
VIDEOEXPOSE	nessuna
USEREVENT	codice

Esempio

Per gestire i nostri eventi, eseguiamo semplicemente il ciclo della coda, controlliamo di che tipo si

tratta (con l'aiuto delle costanti predefinite nel modulo pygame) e quindi eseguiamo qualche azione. Questo codice controlla se l'utente ha premuto il pulsante di chiusura nell'angolo superiore del display e, in caso affermativo, termina il programma.

```
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
```

ATTENZIONE : devi chiamare regolarmente la coda degli eventi quando usi pygame! Oltre al recupero degli eventi disponibili, chiamare la coda degli eventi è anche il modo in cui pygame interagisce internamente con il sistema operativo. Se la coda degli eventi non viene chiamata regolarmente, il sistema operativo supporrà che il tuo programma non funzioni più correttamente e che probabilmente assomigli al programma in crash (in Windows la finestra diventa bianca). Se non vuoi fare nulla con gli eventi dovresti chiamare `pygame.event.pump()` ogni loop di gioco per far processare internamente gli eventi da pygame.

Eventi della tastiera

Esistono due tipi di eventi chiave in pygame: `KEYDOWN` e `KEYUP` . Questi eventi hanno una `key` attributo che è un numero intero che rappresenta un tasto sulla tastiera. Il modulo pygame ha costanti integer predefinite che rappresentano tutte le chiavi comuni. Le costanti sono denominate con la `K` maiuscola, un trattino basso e il nome della chiave. Ad esempio, `<-` è denominato `K_BACKSPACE` , `a` è denominato `K_a` e `F4` è denominato `K_F4` .

Esempio

Questo codice controllerà se l'utente ha premuto `w` , `a` , `s` o `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!")
```

modificatori

Non esiste una costante intera per lettere maiuscole. Invece, gli eventi chiave hanno un altro attributo chiamato `mod` , che è i modificatori (`shift` , `ctrl` , `alt` ecc.) Che vengono premuti simultaneamente come chiave. L'attributo `mod` è un numero intero che rappresenta il modificatore

che viene premuto. Il valore intero di ogni modificatore è memorizzato nel modulo pygame sotto il nome di `KMOD_` e il loro nome. Ad esempio, lo `spostamento a sinistra` è denominato `KMOD_LSHIFT` , `Tab` è denominato `KMOD_TAB` e `Ctrl` è denominato `KMOD_CTRL` .

Esempio

Questo codice controllerà se l'utente ha premuto `a` , `Maiusc` + `a` o `Maiusc` + `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.")
```

Eventi del mouse

Esistono tre tipi di eventi del mouse in `MOUSEMOTION` , `MOUSEBUTTONDOWN` e `MOUSEBUTTONUP` . Pygame registrerà questi eventi quando è stata impostata una modalità di visualizzazione.

`MOUSEMOTION` viene ricevuto quando l'utente sposta il mouse sul display. Ha i `buttons` degli attributi, `pos` e `rel` .

- *`buttons`* sono una tupla che rappresenta se i pulsanti del mouse (`left` , `mouse-wheel` , `right`) vengono premuti o meno.
- *`pos`* è la posizione assoluta (`x` , `y`) del cursore in pixel.
- *`rel`* è la posizione relativa alla posizione precedente (`rel_x` , `rel_y`) in pixel.

`MOUSEBUTTONDOWN` e **`MOUSEBUTTONUP`** vengono ricevuti quando l'utente preme o rilascia un pulsante del mouse. Hanno gli attributi `button` e `pos` .

- *`button`* è un numero intero che rappresenta il pulsante premuto. `1` per il tasto sinistro, 2 per la rotella del mouse e `3` per il tasto destro.
- *`pos`* è la posizione assoluta del mouse (`x` , `y`) quando l'utente preme il pulsante del mouse.

Esempio

Ecco un breve esempio che utilizza alcuni degli attributi di ciascun evento del mouse:

```
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")
```

Poiché non ci sono costanti predefinite per l'attributo del pulsante del mouse nel modulo pygame, ecco i valori per ciascuno:

Pulsante	Valore
Tasto sinistro del mouse	1
Pulsante della rotellina del mouse	2
Pulsante destro del mouse	3
Rotellina del mouse scorrere verso l'alto	4
Rotellina del mouse scorrere verso il basso	5

Scorrendo il pulsante del mouse verranno generati `pygame.MOUSEBUTTONUP` eventi `pygame.MOUSEBUTTONDOWN` e `pygame.MOUSEBUTTONUP` .

Controllo dello stato

È possibile chiamare le funzioni dal modulo `pygame.key` e `pygame.mouse` per ricevere lo stato della chiave e del mouse. Tuttavia, non è il modo consigliato di processare gli eventi in pygame dato che ci sono alcuni difetti con esso:

- Riceverai gli stati quando viene chiamata la funzione, il che significa che potresti perdere gli eventi tra una chiamata e l'altra se l'utente sta premendo rapidamente i pulsanti.
- Non è possibile determinare l'ordine degli eventi.
- Devi comunque chiamare una delle funzioni di evento di pygame per pygame per interagire internamente con il sistema operativo, altrimenti avviserà che il programma non ha risposto. Le funzioni che puoi chiamare sono:
    - `pygame.event.get()` per ottenere tutti gli eventi o tipi di eventi (passando i tipi come argomento) dalla coda.
    - `pygame.event.poll()` per ottenere un singolo evento dalla coda.
    - `pygame.event.wait()` per attendere un singolo evento dalla coda.
    - `pygame.event.clear()` per cancellare tutti gli eventi nella coda.

- `pygame.event.pump()` per consentire a pygame di gestire azioni interne (è chiamato implicitamente dalle funzioni sopra).

Eventi della tastiera

Il modulo chiave ha una funzione `pygame.key.get_pressed()` che restituisce una lista dello stato di tutte le chiavi. L'elenco contiene `0` per tutti i tasti che non sono premuti e `1` per tutti i tasti premuti. Il suo indice nella lista è definito da costanti nel modulo pygame, il tutto preceduto da `K_` e il nome della chiave.

```
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()
```

Se si desidera controllare la pressione di un singolo tasto invece di tenere premuto il tasto, è possibile memorizzare lo stato precedente di tutte le chiavi in una variabile temporanea e verificare se il valore cambia:

```
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'istruzione viene valutata su true solo quando il tasto corrente viene premuto e il tasto precedente non viene premuto. Per verificare se l'utente ha rilasciato la chiave devi solo cambiare la parola chiave `not` (`if not key[pygame.K_q] and previous_key[pygame.K_q]`). Affinché funzioni correttamente, devi impostare la variabile `previous_key = pygame.key.get_pressed()` prima del ciclo di gioco, altrimenti riceverai un `NameError` .

Eventi del mouse

Il modulo del mouse ha funzioni che ci permettono di controllare e impostare la posizione del mouse e di controllare i pulsanti premuti. La funzione `pygame.mouse.get_pressed()` restituisce una tupla di tupla che rappresenta se i pulsanti del mouse (sinistra, rotellina del mouse, destra) vengono premuti o meno.

```
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()
```

Leggi Manipolazione degli eventi online:
https://riptutorial.com/it/pygame/topic/5110/manipolazione-degli-eventi

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con pygame	Community, elegent, Inazuma, Nearoo, Ni., numbermaniac, svs, Ted Klein Bergman, White Shadow
2	Aggiunta di musica di sottofondo e effetti sonori	Hikaryu, White Shadow
3	Creare una finestra in pygame - pygame.display.set_mode ()	Nearoo
4	Creare una semplice finestra pygame	ModoUnreal
5	Creazione di una finestra Pygame	Rishi Malhotra, White Shadow
6	Disegnare sullo schermo	svs
7	L'essenziale	Ni., numbermaniac, svs, Ted Klein Bergman
8	Manipolazione degli eventi	elegent, Mikhail V, Nearoo, Ted Klein Bergman