Turtle

La libreria Turtle di Python non ha moduli separati nel senso tradizionale, ma è un singolo modulo con diverse funzioni e classi che permettono di creare grafica vettoriale.
Tuttavia, all'interno del modulo turtle, si possono identificare diverse categorie di funzionalità:

- **Controllo dello schermo**: Funzioni per controllare la finestra di disegno (cambiare il colore di sfondo, cancellare il disegno, ecc..)
- **Movimento della tartaruga (penna)**: Funzioni per spostare la tartaruga (avanti, indietro, gira a destra, gira a sinistra, ecc.).
- **Controllo della tartaruga (penna)**: Funzioni per alzare e abbassare la penna, cambiare il colore della penna, cambiare lo spessore della penna, ecc.
- **Controllo del riempimento**: Funzioni per riempire le forme con un colore.
- **Controllo e scrittura del testo** : funzioni per scrivere del testo
- **Eventi**:funzioni per il controllo della pressione dei tasti o click del mouse
- **Altre funzioni**: Funzioni per disegnare cerchi, ecc.

Inoltre, esiste un modulo chiamato turtledemo che fornisce esempi di come utilizzare la libreria Turtle. Questo modulo non fa parte della libreria Turtle principale, ma è incluso nella distribuzione standard di Python.
In sintesi, la libreria Turtle è un singolo modulo con molte funzioni, e c'è anche un modulo di dimostrazione separato.

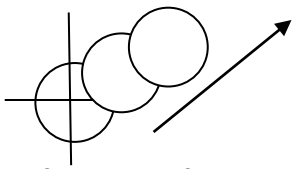
Esempio:

```
import turtle

# Crea un oggetto tartaruga
tartaruga = turtle.Turtle()

# Disegna un quadrato
for _ in range(4):
    tartaruga.forward(100)  # Muovi la tartaruga in avanti di 100 pixel
    tartaruga.left(90)     # Ruota la tartaruga di 90 gradi a sinistra

# Chiudi la finestra grafica quando si clicca su di essa
turtle.exitonclick()
```

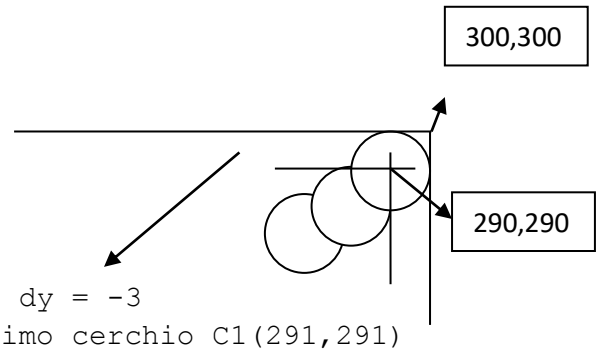
Animazioni

1. Movimento:

Esempio



```
dx = 3 e dy =3
centro primo cerchio C1(0,0)
centro secondo cerchio C2(3,3)
centro terzo cerchio C3(6,6)
```




```
dx = -3 e dy = -3
centro primo cerchio C1(291,291)
centro secondo cerchio C2(288,288)
centro terzo cerchio C3(285,285)
```

Animazione di una pallina che si muove nello schermo usando il modulo turtle di Python. Ecco il codice:

```
import turtle
import time

# Imposta lo schermo
schermo = turtle.Screen()
schermo.bgcolor("white")
schermo.setup(width=600, height=600)
schermo.tracer(0)  # Disabilita l'aggiornamento automatico dello schermo

# Crea la pallina
pallina = turtle.Turtle()
pallina.shape("circle")
pallina.color("red")
pallina.penup()
pallina.goto(0, 0)
pallina.dx = 3  # Velocità orizzontale
pallina.dy = 3  # Velocità verticale

# Funzione per muovere la pallina
def muovi_pallina():
  pallina.setx(pallina.xcor() + pallina.dx)
  pallina.sety(pallina.ycor() + pallina.dy)

  # Controllo dei bordi
  if pallina.xcor() > 290 or pallina.xcor() < -290:
    pallina.dx *= -1  # Inverti la direzione orizzontale
  if pallina.ycor() > 290 or pallina.ycor() < -290:
    pallina.dy *= -1  # Inverti la direzione verticale

# Ciclo principale del gioco
while True:
  schermo.update()  # Aggiorna lo schermo
  muovi_pallina()
  time.sleep(0.01)  # Pausa per controllare la velocità dell'animazione
```

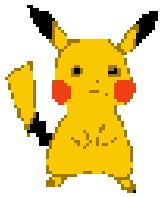
2. Sprite ed eventi

In informatica, uno **sprite (personaggio)** è un'immagine bidimensionale (2D) che fa parte di una scena più grande e che può essere spostata in modo indipendente sullo schermo.

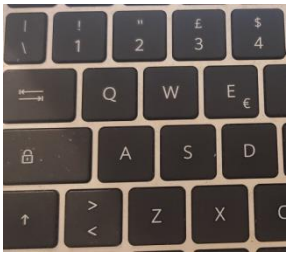
Caratteristiche principali:

- **Mobilità:** Gli sprite possono essere spostati, ruotati e ridimensionati liberamente all'interno di una scena.
- **Sovrapposizione:** Gli sprite possono essere sovrapposti ad altri elementi grafici, come sfondi o altri sprite.
- **Animazione:** Gli sprite possono essere animati, cambiando rapidamente le immagini per creare l'illusione del movimento.

Efficienza: L'uso degli sprite può migliorare le prestazioni grafiche, poiché è più efficiente spostare un'immagine predefinita che ridisegnare l'intera scena.



Eventi



In Python, il modulo turtle consente di gestire gli eventi della tastiera per creare interazioni con l'utente. Ecco come funzionano e un esempio pratico:

Gestione degli eventi della tastiera con turtle

Funzioni di callback:
Si definiscono funzioni che verranno eseguite quando vengono premuti determinati tasti. Queste funzioni sono chiamate "callback".

Associazione degli eventi:
Si utilizza il metodo onkey() dell'oggetto turtle.Screen() per associare le funzioni di callback ai tasti specifici.

Ascolto degli eventi:
Il metodo listen() dell'oggetto turtle.Screen() avvia l'ascolto degli eventi della tastiera.

Esempio pratico

```
import turtle

# Funzione per spostare la tartaruga in avanti
def avanti():
    tartaruga.forward(50)

# Funzione per spostare la tartaruga indietro
def indietro():
    tartaruga.backward(50)

# Funzione per ruotare la tartaruga a sinistra
def sinistra():
    tartaruga.left(45)

# Funzione per ruotare la tartaruga a destra
def destra():
    tartaruga.right(45)

# Crea una finestra e una tartaruga
finestra = turtle.Screen()
tartaruga = turtle.Turtle()

# Associa le funzioni ai tasti
finestra.onkey(avanti, "w")
finestra.onkey(indietro, "s")
finestra.onkey(sinistra, "a")
finestra.onkey(destra, "d")

# Inizia l'ascolto degli eventi
finestra.listen()
```

```
# Mantiene la finestra aperta
finestra.mainloop()
```

Spiegazione dell'esempio

- Vengono definite quattro funzioni: avanti(), indietro(), sinistra() e destra(), che spostano e ruotano la tartaruga.
- Viene creata una finestra finestra e una tartaruga tartaruga.
- Il metodo onkey() viene utilizzato per associare ciascuna funzione a un tasto specifico (w, s, a, d).
- Il metodo listen() permette di intercettare gli eventi della tastiera.
- Il metodo mainloop() mantiene la finestra aperta, permettendo all'utente di interagire con la tartaruga usando i tasti associati.

In questo modo, premendo i tasti "w", "s", "a" o "d", la tartaruga si muoverà di conseguenza.

3. Scrittura testo (punteggio)

Punteggio : introduciamo tre variabili globali

```
aggiornaPunteggio = False
aggiornaPunteggioOld = False
punteggio = 0

class Penna(turtle.Turtle):
    def __init__(self, x, y, colore, dimensione, carattere):
        super().__init__()
        self.x = x
        self.y = y
        self.colore = colore
        self.dimensione = dimensione
        self.carattere = carattere
        self.up() #alzo la penna dal foglio
        #self.scrivi("ciao mondo")

    def scrivi(self, testo):
        self.up()
        self.goto(self.x , self.y)
        print("ciao")
        self.down()
        self.write(testo, font = (self.carattere, self.dimensione, "normal"))
        self.up()
        #self.hideturtle()
```

https://github.com/albertocesaretti/Python_Turtle_Pygame_Tkinter