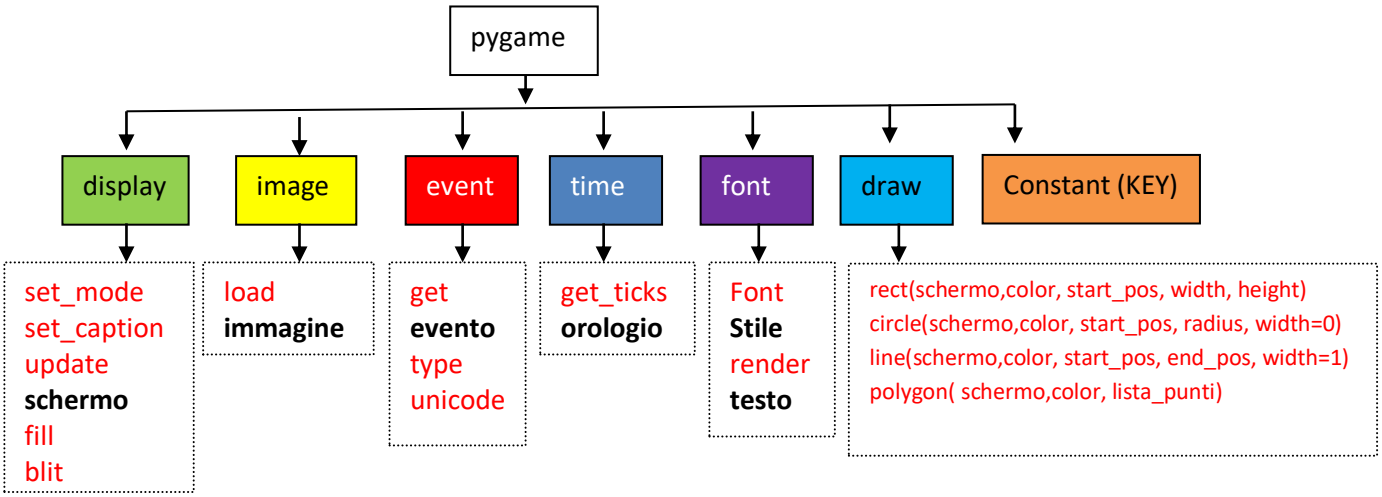
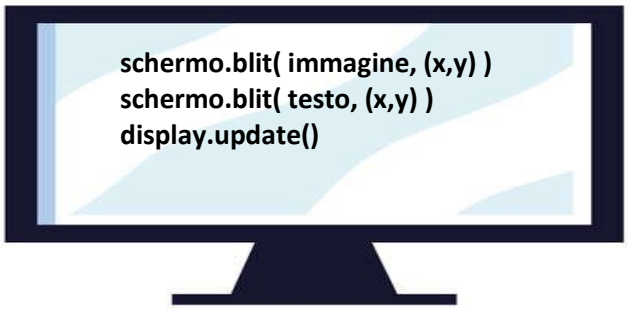
https://github.com/albertocesaretti/

PYGAME

Moduli:

1. Cos'è pygame e sua storia
2. Il modulo **display**
3. Il modulo **image**
4. Il modulo **event**
5. Il modulo **time**
6. Il modulo **font**
7. Il modulo **draw**
8. Il modulo **constant**





La differenza tra la funzione display.update e la funzione display.flip.
La funzione **update può aggiornare una porzione di** schermo mentre la funzione flip aggiorna sempre tutto lo schermo

1. Cos'è pygame

Pygame è una libreria dedicata allo sviluppo di giochi: contiene funzioni per la grafica, il suono, il movimento delle immagini e così via. Per usare queste funzioni dovremo quindi installarla sulla nostra distribuzione di Python.

Storia

Origini e sviluppo:

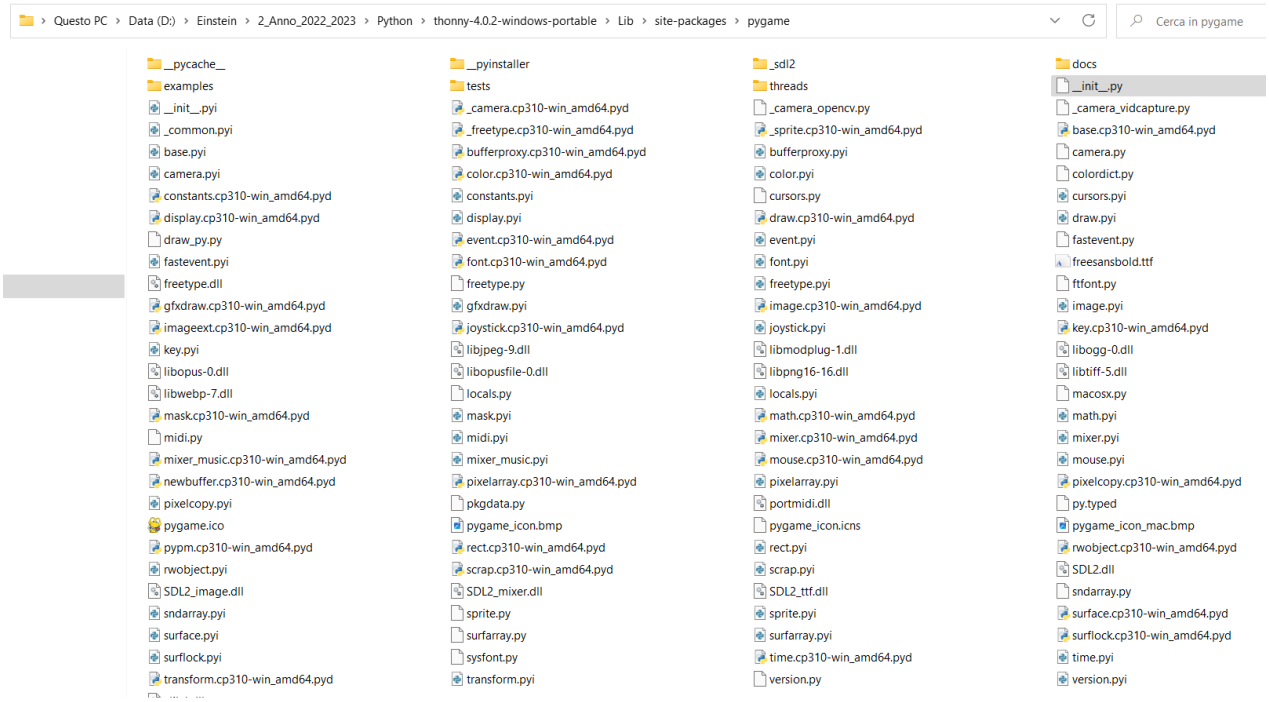
- **SDL (Simple DirectMedia Layer):** Pygame si basa sulla libreria SDL, scritta in C da Sam Lantinga. SDL fornisce l'accesso a basso livello all'hardware audio, tastiera, mouse, joystick e grafica 2D tramite OpenGL.
- **Creazione di Pygame:** Sulla base di SDL, è stato creato Pygame per fornire un'interfaccia Python per queste funzionalità, rendendo più semplice per gli sviluppatori Python la creazione di giochi.
- **Prima versione:** La prima versione di Pygame è stata rilasciata il 28 ottobre 2000.
- **Open source:** Pygame è un progetto open source, il che significa che è gratuito da usare e modificare. È rilasciato sotto la licenza LGPL (GNU Lesser General Public License).

Comunità e sviluppo continuo: Pygame ha una comunità attiva di sviluppatori che contribuiscono al suo sviluppo e miglioramento

Dato che Pygame è una libreria piuttosto consistente che si occupa di molti elementi diversi (grafica, suono, input/output ...) i suoi programmatori hanno deciso di dividerlo a sua volta in ***sottomoduli***, cioè in files separati nei quali sono raggruppate le funzioni che gestiscono determinati elementi.

Nella terminologia di Python questo tipo di libreria divisa in sottomoduli si chiama *package*. Per capire la struttura di Pygame ci conviene tenere aperta la pagina ufficiale della documentazione:

www.pygame.org/docs



display	contiene funzioni per impostare le proprietà della finestra principale
image	contiene funzioni per caricare e visualizzare immagini
font	contiene funzioni per impostare lo stile del testo
key	contiene funzioni per l'input da tastiera
draw	contiene funzioni per disegnare sullo schermo
event	contiene funzioni per gestire gli eventi
time	Contiene funzioni per gestire orologi o timer
cursor	contiene funzioni per modificare il cursore del mouse
mixer	contiene funzioni per caricare e suonare files sonori

2. Il modulo display per creare lo schermo

Creazione di una finestra con schermo nero di dimensioni prefissata in pixel

```
1. import pygame                                    #importo la libreria
2. pygame.init( )                                   #inizializzo pygame
3. schermo = paygame.display. set_mode( (800,600))  # crea lo schermo
4. pygame.display.set_caption("hello world")        #cambio il titolo della finestra
5.  # pygame.display.set_icon("image.ico")          #cambio l'icona della finestra
6. schermo.fill((255,0, 0))                         #coloro di rosso lo schermo
7. pygame.display.update( )                          #aggiorno una porzione di finestra
8. pygame.display.flip()                        #aggiorno l'intero contenuto della finestra
```

3. Il modulo image

Per caricare le immagini nell'area di lavoro si usa il modulo image , con la funzione load,seguendo la seguente sintassi. Alla funzione load viene passato il nome del file che contiene l'immagine,
come da esempi:

```
9. gatto = pygame.image.load('gatto.jpg')
10. cane = pygame.image.load('cane.jpg')
11. icona = pygame.image.load('batteria.png')
```

I file immagine possono avere vari formati.
Le immagini vengono caricate nello spazio di lavoro di pygame ma ancora non visualizzate sullo schermo

Lo schermo poi deve copiare le immagini sullo schermo tramite la funzione blit che significa riprodurre

```
12. schermo.blit(gatto, (0,0))          #copia l'immagine del gatto nell'origine
13. pygame.display.update( )            # aggiorno poi il display
```

4. Il modulo event:

Gli eventi

Per ora siamo riusciti a creare una finestra e a disegnare qualcosa su di essa. Per cominciare ad interagire con il nostro programma è necessario conoscere meglio il funzionamento di un'interfaccia grafica.
Windows, Mac, Linux o il nostro smartphone usano delle interfacce grafiche (in inglese GUI: Graphic User Interface) tra l'utente ed il computer: in un'applicazione moderna quasi tutta l'interazione tra l'uomo ed il computer avviene muovendo il mouse, cliccando su immagini, bottoni, menu, ecc.
Come funziona di fatto tutto questo?
Bene, mentre noi lavoriamo al computer il sistema operativo controlla continuamente tutto quello che avviene: la tastiera, il cursore del mouse, i pulsanti del mouse, ecc. Ogni volta che accade qualcosa (muoviamo il mouse, facciamo clic, premiamo un tasto della tastiera, apriamo una nuova finestra...) esso genera un **evento di sistema, cioè un messaggio che contiene informazioni su quello che è accaduto, e lo invia a tutte le applicazioni aperte in quel momento**.
Ogni applicazione che lo riceve può analizzarlo, capire che cosa è successo e comportarsi di conseguenza (ad esempio un clic sul bottone "Chiudi" provocherà la chiusura di una finestra, ecc). Poichè gli eventi possono susseguirsi anche in maniera molto rapida esiste una coda degli eventi per ogni applicazione, alla quale vengono spediti gli eventi in ordine di arrivo (come le persone all'ufficio postale); le applicazioni li esaminano uno dopo l'altro decidendo cosa fare (possono anche ignorare quelli che non interessano).
Ad esempio, un effetto ormai comune in tutte le applicazioni è l'evidenziazione delle varie icone quando il mouse vi passa sopra (lo faremo come esercizio nel prossimo capitolo). Ogni volta che il mouse si muove, il sistema operativo genera un evento che contiene le coordinate del puntatore e lo invia alle applicazioni aperte in quel momento. Il programma lo riceve, fa qualche calcolo e rileva se il puntatore è entrato nel rettangolo dove è

disegnata l'icona; se la risposta è sì cambia l'immagine dell'icona con quella dell'icona evidenziata. Allo stesso modo, quando il puntatore esce dal rettangolo, cambia nuovamente l'immagine in quella normale.

GLI EVENTI IN PYGAME

Vediamo infine come fare per ricevere gli eventi dal sistema operativo. Il sottomodulo event di pygame contiene alcune funzioni per gestire gli eventi che il sistema operativo trasmette al nostro programma. Abbiamo detto che al nostro programma è assegnata una *coda di eventi* (per noi invisibile perchè gestita da Windows): la funzione **pygame.event.get()**
prende gli eventi dalla coda, li trasforma in oggetti Event di pygame e ci restituisce una lista di Python che contiene i nostri Event in ordine cronologico. Ecco un esempio molto semplice dell'uso di questa funzione:

```
14.  for event  in pygame.event.get():
        print("ho ricevuto l'evento", event)
        if (event.type == pygame.KEYDOWN):  #tasto premuto
            if event.unicode == '1':
                print('digitato 1 sulla tastiera')
```

5. Il modulo time

L'oggetto più usato del modulo è comunque il Clock, un vero orologio che misura i millesimi di secondo. Questo oggetto ha un metodo tick() che rappresenta un "battito" dell'orologio. Deve essere chiamato con un argomento che rappresenta il *frame rate* (FPS) che si vuole ottenere: ogni volta che viene chiamato calcola il tempo trascorso dall'ultimo tick() ed arresta il programma per il tempo necessario ad ottenere la temporizzazione. **Useremo il tick() come *ultima istruzione* del ciclo principale**: dopo aver svolto tutti i compiti richiesti Python aspetterà il tempo appropriato prima di ricominciare il ciclo, ottenendo così una temporizzazione esatta. Vediamo un semplice esempio:

```
import pygame
pygame.init()
schermo = pygame.display.set_mode((800, 600))
# creiamo un orologio
orologio = pygame.time.get_ticks()
```

6. Il modulo draw

Il sottomodulo draw di pygame contiene alcune funzioni che servono a disegnare su una Surface. Per ogni sottomodulo la documentazione ufficiale di pygame contiene una pagina dove sono elencate e spiegate (in Inglese) tutte le funzioni definite in esso, che vi consiglio di tenere sempre d'occhio per riferimento, anche se non capite tutto quello che c'è scritto. Questo è il link alla documentazione del modulo draw, del quale vediamo ora alcune funzioni.
Ad esempio la funzione circle() è così definita:

```
circle(Surface, color, pos, radius, width=0) -> Rect.
```

La funzione serve e disegnare un cerchio e questo è il significato dei parametri:

NUOVO PROGRAMMA: prova_surface.py

```
import pygame

# inizializza pygame: chiamare sempre per prima
pygame.init()

# finestra principale (prende come parametro una tupla con le dimensioni)
```

```
screen = pygame.display.set_mode((800, 600))

pygame.draw.circle(screen, "red", (200, 200), 120)

# aggiorna lo schermo dopo che abbiamo disegnato su screen
pygame.display.flip()

#termina correttamente il programma
input("Premi INVIO per terminare il programma")
pygame.quit()
```

Vediamo ora la funzione line(), che serve a tracciare un segmento:
line(schermo, color, start_pos, end_pos, width=1) -> Rect.

```
pygame.draw.line(screen, "green", (400, 100), (600, 500), 4)
```

vedrete ora anche un segmento verde piuttosto spesso.
Un po' più difficili da usare sono le funzioni draw.polygon() e draw.lines() che prendono come parametro una lista (o tuple) di punti.

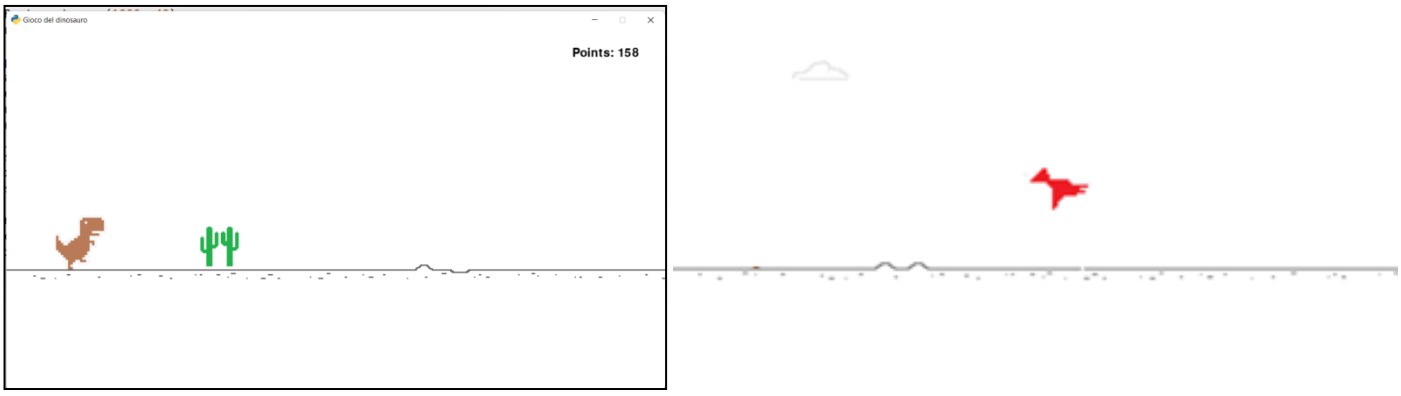
```
lista_punti = ((100, 400), (250, 600), (350, 550), (400, 450), (200, 400))
pygame.draw.polygon(screen, "yellow", lista_punti)
```

Le altre funzioni sono: rect(), ellipse(), arc()

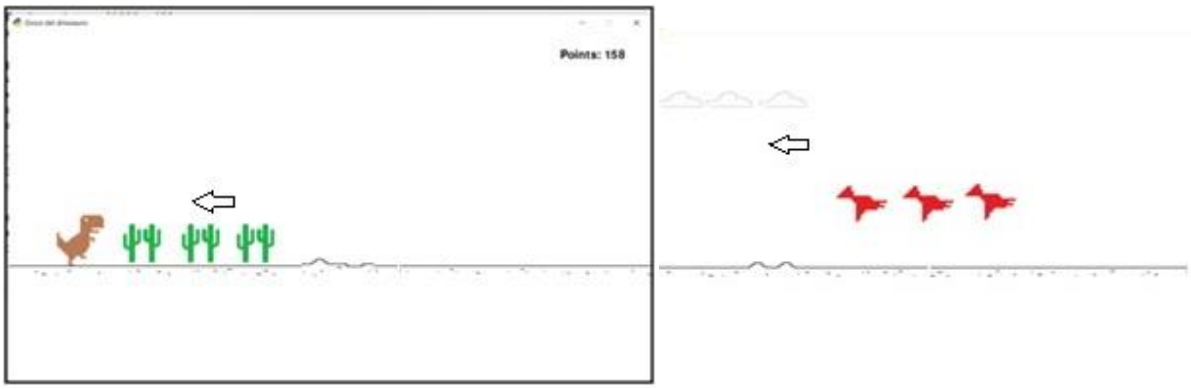
Gioco del dinosauro

Scene e disegno:

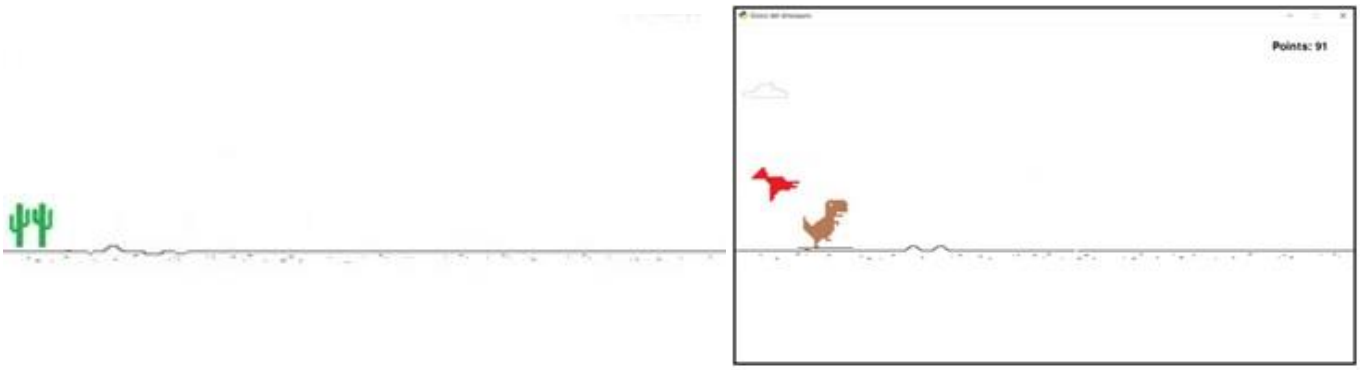
Vengono disegnate due scene una nello schermo e l'altra fuori schermo a dx



Ad ogni ciclo gli sprite vengono cancellati e ridisegnati in una posizione di 20 pixel in x (game_speed = 20), più a sinistra in modo che si avvicinino al dinosauro

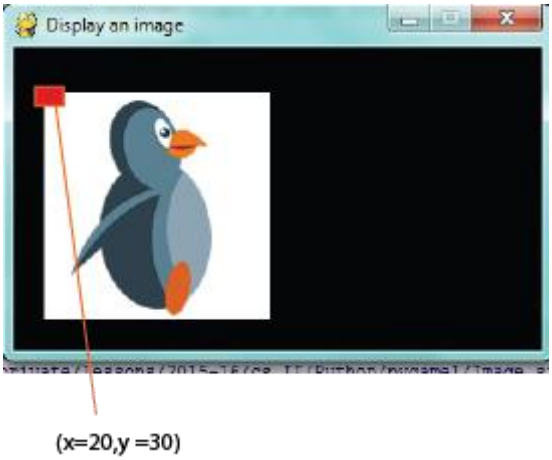


In questa immagine sotto si vede che la prima scena esce dallo schermo scorrendo a sinistra ed entra la seconda scena che prima non appariva nello schermo

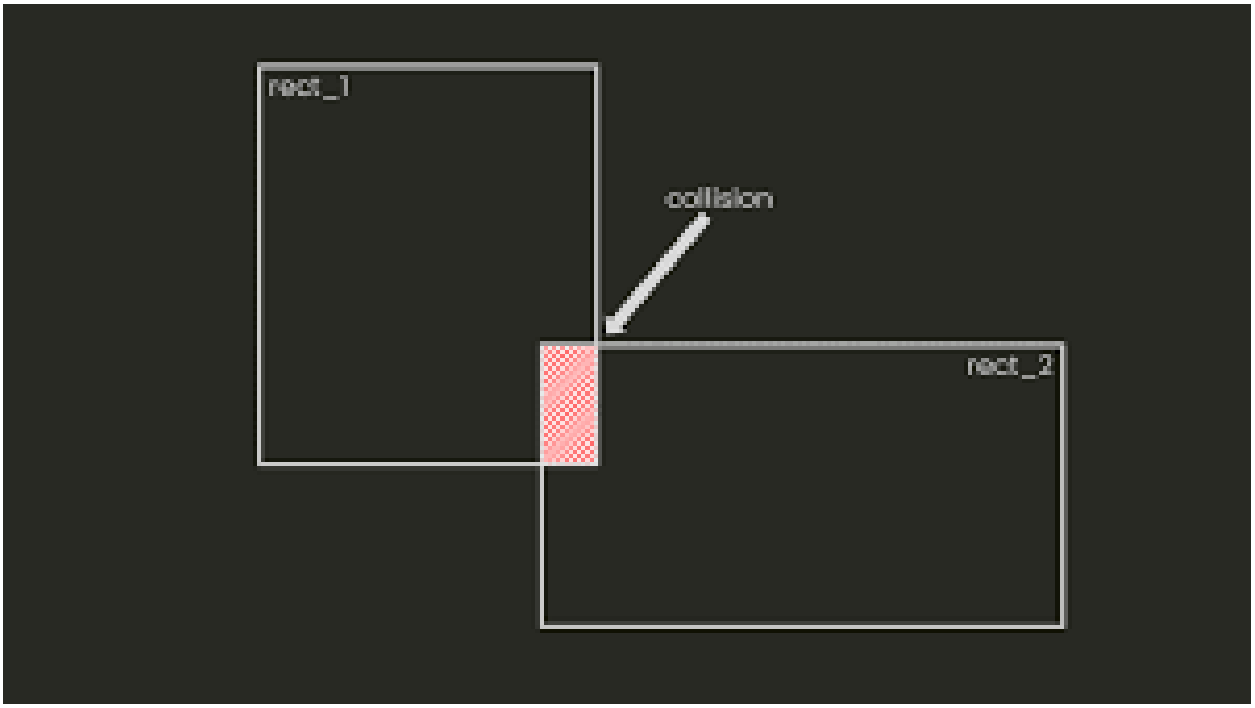


Collisioni

Ogni sprite ha associato un rettangolo invisibile, di coordinate x, y e larghezza altezza data dalle dimensioni dell'immagine



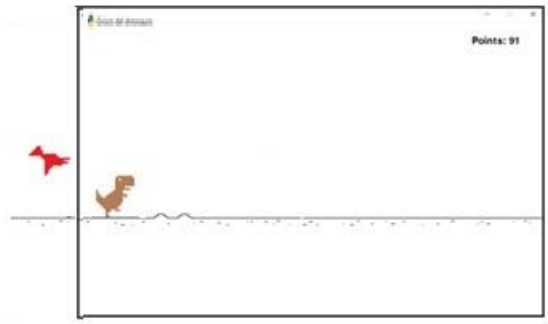

```
self.pinguino_rect = self.image.get_rect()
self.pinguino_rect.x = self.X_POS
self.pinguino_rect.y = self.Y_POS
```



Quando due rettangoli di due sprite si sovrappongono in parte si ha una collisione

Lista ostacoli

L'ostacolo viene rimosso dalla lista quando esce di scena



Confronto tra le tre librerie

Una panoramica delle differenze tra le librerie grafiche Python Turtle, Tkinter e Pygame:

Turtle

- **Scopo principale:**
  - È progettata principalmente per scopi educativi, per introdurre i principi di base della programmazione grafica.
  - È utile per creare disegni geometrici e semplici animazioni.
- **Caratteristiche:**
  - Interfaccia semplice e intuitiva, ideale per i principianti.
  - Utilizza un "turtle" (una tartaruga) che si muove sullo schermo, lasciando una traccia.
  - Fornisce funzioni per controllare il movimento, la direzione e il colore della tartaruga.
- **Limiti:**
  - Limitata in termini di grafica avanzata e interattività.
  - Non è adatta per lo sviluppo di giochi complessi o applicazioni grafiche professionali.

Tkinter

- **Scopo principale:**
  - È una libreria standard per la creazione di interfacce grafiche utente (GUI) in Python.
  - È adatta per lo sviluppo di applicazioni desktop con finestre, pulsanti, menu e altri elementi interattivi.
- **Caratteristiche:**
  - Fornisce una vasta gamma di widget (elementi GUI) per creare interfacce utente personalizzate.
  - È integrata nella libreria standard di Python, quindi non richiede installazioni aggiuntive.

    - È multipiattaforma, il che significa che le applicazioni Tkinter possono essere eseguite su diversi sistemi operativi.
- **Limiti:**
    - La grafica può sembrare meno moderna rispetto ad altre librerie GUI.
    - Potrebbe non essere la scelta migliore per giochi con grafica 3D o animazioni complesse.

Pygame

- **Scopo principale:**
    - È progettata specificamente per lo sviluppo di videogiochi 2D in Python.
    - Fornisce funzionalità per la gestione di grafica, audio, input e eventi.
- **Caratteristiche:**
    - Offre un controllo preciso sulla grafica e sulle animazioni.
    - Supporta la gestione di sprite, collisioni, suoni e musica.
    - Fornisce strumenti per la gestione dell'input da tastiera, mouse e joystick.
- **Limiti:**
    - Richiede una maggiore curva di apprendimento rispetto a Turtle o Tkinter.
    - Potrebbe richiedere l'installazione di librerie aggiuntive per alcune funzionalità avanzate.

In sintesi:

- **Turtle:** Ideale per l'apprendimento dei concetti di base della grafica e della programmazione.
- **Tkinter:** Ottima per la creazione di applicazioni desktop con interfacce utente semplici e funzionali.
- **Pygame:** La scelta migliore per lo sviluppo di videogiochi 2D con grafica e animazioni avanzate.

Spero che questa panoramica ti sia utile!