



PyGame Zero

Ovvero: come diventare game developer senza impazzire

Recap: "Colpisci l'alieno"

Per chi era distratto (sappiamo chi siete 🙄)

 III Liceo Scientifico Biella - Scienze Applicate

 Python Biella Group

P
BG



Ma cos'è PyGame Zero?

PyGame Zero è la versione "Easy Mode" di PyGame, ovveo una libreria pensata per **rendere facile** creare videogiochi 2D in Python.

```
# PyGame normale
import pygame
pygame.init()
screen = pygame.display.set_mode((800, 600))
clock = pygame.time.Clock()
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
# ... e siamo solo all'inizio 😬
```




PyGame Zero: il superpotere

```
import pgzrun

# BAM! Finestra pronta
WIDTH = 800
HEIGHT = 600

def draw():
    screen.fill('blue') # Fatto!

pgzrun.go()
```

In pratica: PyGame Zero fa la magia dietro le quinte, fa la parte noiosa per te.

Tu: scrivi poche funzioni e ti concentri sul gioco (la parte divertente).

E' come avere una PlayStation con i cheat code già attivati.



Cosa nasconde PyGame Zero?

Dietro le quinte, PyGame Zero è come un maggiordomo inglese:

- Crea la finestra automaticamente
- Gestisce il game loop (il ciclo infinito del gioco)
- **Controlla** eventi di mouse e tastiera
- Disegna lo schermo 60 volte al secondo
- Carica immagini e suoni senza che tu debba chiedere

Traduzione: Fa tutto il lavoro sporco mentre tu ti prendi i complimenti! 😎



Il Game Loop: il cuore pulsante

Ogni gioco ha un ciclo che si ripete in eterno:

1. Cattura input utente
2. Aggiorna stato gioco
3. Disegna tutto
4. Aspetta un pochino



(Ripeti forever)

PyGame Zero nasconde questo loop!

Tu scrivi solo le funzioni `draw()` e `update()`.



Nel codice compaiono alcune costanti importanti:

```
TITLE = "Colpisci l'alieno"  
WIDTH = 800  
HEIGHT = 600
```

Servono a:

- **TITLE** → nome della finestra.
- **WIDTH/HEIGHT** → dimensioni del gioco (in pixel).
- PyGameZero le legge automaticamente per creare la finestra corretta.



Perché MAIUSCOLE?

È la convenzione Python per dire: "*NON MI TOCCARE!*" 🚫

Sono valori che:

- Si definiscono **una volta**
- Non cambiano **mai** durante il programma
- Hanno nomi **GRIDATI** per farsi rispettare



Domanda: Perché non scrivere direttamente 800 nel codice?

Brutto ❌

```
screen.draw.text(messaggio, center=(400, 40))  
alieno.x = randint(50, 750)
```

Bello ✅

```
screen.draw.text(messaggio, center=(WIDTH/2, 40))  
alieno.x = randint(50, WIDTH-50)
```

Risposta: Se domani vuoi finestra 1024x768, cambi 1 riga invece di 2,10,47 righe!



La funzione `draw()`: l'artista

```
def draw():  
    screen.clear()           # Cancella tutto  
    screen.fill(color=(128, 0, 0)) # Sfondo rosso scuro  
    alieno.draw()           # Disegna l'alieno  
    screen.draw.text(messaggio, ...) # Scrivi il testo
```

Chiamata automaticamente 60 volte al secondo!

È come un pittore ossessivo-compulsivo che ridipinge la tela continuamente.



La funzione `draw()`

È LA star del programma.

- PyGameZero la chiama **automaticamente** ogni volta che deve ridisegnare lo schermo.
- Serve per mostrare:
 - sfondi
 - sprite
 - testi
 - esplosioni epiche

In breve:

*Se non lo metti in `draw()` , **non apparirà mai.**
Fine della storia.*



draw(): le regole d'oro

1. L'ordine conta!

```
alieno.draw()      # Alieno sopra  
screen.fill('red') # Sfondo copre tutto ❌
```

2. Viene chiamata DA SOLA (non devi fare draw())

3. È VELOCE: eseguita 60 volte/sec, non fare calcoli pesanti qui!



Il modulo `clock`: il metronomo

```
from pgzero.clock import clock  
  
clock.schedule_interval(piazza_alieno, 1.0)
```

Traduzione umana:

"Ehi clock, chiama piazza_alieno() ogni 1 secondo, grazie!"

È come impostare un timer, ma per funzioni!



Il modulo `clock`

Il modulo `clock` permette di programmare azioni che accadono nel tempo.

Nel gioco:

- “Ehi clock, ogni secondo sposta l’alieno da qualche parte!”
- È come una sveglia che suona ogni 1.0 secondi.
- Perfetto per: timer, nemici che si muovono, animazioni, effetti...

🔧 *Dietro le quinte, PyGameZero usa un sistema di **eventi temporizzati**: elegante, potente e soprattutto... non dovete programmarlo voi!* 🎉



clock: i superpoteri

```
# Chiama funzione ogni N secondi  
clock.schedule_interval(funzione, secondi)  
  
# Chiama funzione UNA VOLTA dopo N secondi  
clock.schedule(funzione, secondi)  
  
# Cancella tutto  
clock.unschedule(funzione)
```

Nel nostro gioco: l'alieno si sposta ogni secondo.

Senza clock: dovresti contare i frame a mano (che noia! 🤔)



Il nostro gioco: anatomia

```
alieno = Actor("alieno") # Crea l'attore
```

Actor = personaggio/oggetto del gioco con:

- Immagine (automaticamente da `images/alieno.png`)
- Posizione (`x, y`)
- Dimensioni (`width, height`)
- Collisioni (`collidepoint()`)

È un oggetto con **superpoteri** inclusi!



La funzione piazza_alieno()

```
def piazza_alieno():  
    alieno.x = randint(50, WIDTH-50)  
    alieno.y = randint(50, HEIGHT-50)  
    alieno.image = "alieno"
```

Cosa fa:

- Posizione casuale (ma non troppo vicino ai bordi!)
- Resetta l'immagine (caso fosse "esplosione")

Challenge: Perché 50 e non 0? 🤔

Hint: l'alieno è 64x64 pixel...

P
BG



on_mouse_down(): il detective

Questa funzione intercetta i clic del mouse.

```
def on_mouse_down(pos):  
    global messaggio  
    if alieno.collidepoint(pos):  
        messaggio = "Bel colpo!"  
        alieno.image = "esplosione"  
    else:  
        messaggio = "Mancato..."
```

Chiamata automaticamente quando clicchi!

- pos = coordinate (x, y) del click
- collidepoint() = verifica se hai colpito l'alieno



La keyword global: il ribelle

```
global messaggio
```

Problema: Le funzioni hanno "memoria corta"

Soluzione: `global` dice: *"Voglio modificare la variabile ESTERNA!"*

```
x = 10

def cambia():
    x = 20  # Crea nuova variabile locale ✗

def cambiadavvero():
    global x
    x = 20  # Modifica quella esterna ✓
```




Ricapitoliamo: il flusso del gioco

```
# 1. SETUP (eseguito UNA volta)
alieno = Actor("alieno")
piazza_alieno() # Posizione iniziale
clock.schedule_interval(piazza_alieno, 1.0) # Timer

# 2. GAME LOOP (automatico, 60 FPS)
draw() # ← Disegna tutto

# 3. EVENTI (quando succedono)
on_mouse_down(pos) # ← Click del mouse
```

PyGame Zero orchestra tutto! 



Pro Tips finali

1. **screen**: l'oggetto magico che rappresenta la finestra
2. **Actor**: per qualsiasi cosa si muova/interagisca
3. **clock**: per tutto ciò che è temporizzato
4. **Convenzioni nomi**: `snake_case` per funzioni, `MAIUSCOLE` per costanti
5. Le immagini vanno in `images/`, i suoni in `sounds/`

Documentazione: pygame-zero.readthedocs.io



Domande?

"L'unica domanda stupida è quella non fatta"

(Seguita da vicino da "Come mai non funziona?" senza mostrare il codice) 😊

Ora: Aprite Thonny e sperimentate! 🚀



SFIDA FINALE

Livello 1: Fai apparire l'alieno ogni 0.5 secondi invece di 1

Livello 2: Cambia il colore dello sfondo quando colpisci l'alieno e cambia immagine all'alieno *anche* quando sbagli (magari diventa arrabbiato?)

Livello 3: Aggiungi un contatore di colpi riusciti

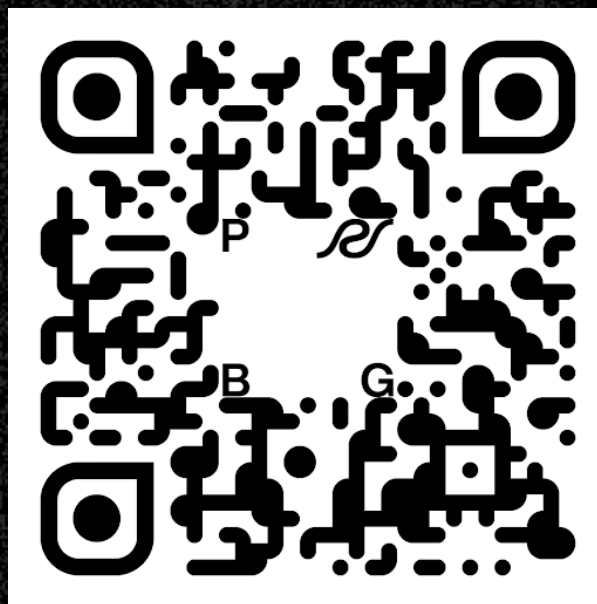
Livello 4: Fai sparire il messaggio dopo 0.5 secondi

Livello Boss: L'alieno si rimpicciolisce ogni colpo!

Chi accetta le sfide? 



Grazie per l'attenzione...



"C'è sempre qualcosa da imparare per migliorarci e crescere...insieme!"