PBG

🪄 Pygame Zero & Polars

⚡ La Battaglia Finale – Harry vs Voldemort

Approfondimento su Data Science e Game Logic

💻 III Liceo Scientifico Biella - Scienze Applicate

Python Biella Group

🎯 Obiettivi Didattici Avanzati

In questo progetto non ci limiteremo a muovere sprite, ma impareremo a:

1. **Gestione Dati Esterna**: Separare il codice dai dati (incantesimi) usando file CSV/Excel.
2. **Motore di Calcolo**: Usare **Polars** per interrogare velocemente le tabelle.
3. **State Machine**: Gestire il passaggio tra turno del giocatore, animazione e turno della CPU.
4. **Feedback Visivo**: Creare animazioni che comunichino al giocatore cosa sta succedendo (danni, fallimenti, vittorie).
5. **User Interface (UI)**: Disegnare menu dinamici basati su coordinate matematiche.

✅ Requisiti

- Python installato
- Librerie: `pgzero`, `polars`
- Cartella del progetto con:
  - `harry_potter.py` : il codice
  - `spells.csv` : il database
  - Cartella `images/` con `harry.png`, `voldemort.png`, `vittoria.png`, `sconfitta.png`

Import e impostazioni

```
import random
import polars as pl
import pgzrun
from pgzero.actor import Actor
from pgzero.clock import clock

WIDTH = 800
HEIGHT = 600
TITLE = "La Battaglia Finale: Harry vs Voldemort"
```

- `random` serve per probabilità
- `polars` legge il CSV (lista di incantesimi)
- `pgzero` è il motore grafico: disegna e gestisce input

Polars: Gestire gli incantesimi

Per la scelta degli incantesimi usiamo un database:

```
import polars as pl
incantesimi_df = pl.read_csv("spells.csv")
def ottieni_opzioni(personaggio):
    return incantesimi_df.filter(pl.col("character") == personaggio)
```

Il file contiene righe con colonne tipo: `character` (Harry o Voldemort), `spell` (nome dell'incantesimo), `damage` (numero — negativo = cura), `precision` (0.0–1.0)

```
Harry,Stupefy,20,0.85
Voldemort,Crucio,30,0.7
```

La Gestione dei Turni (State Control)

Il gioco non avviene tutto insieme. Abbiamo bisogno di variabili che controllino il "flusso":

```
attesa_input = True  # Il gioco aspetta che tu clicchi un incantesimo
gioco_attivo = True  # La partita è in corso o qualcuno è stato sconfitto?
```

- **Turno Harry**: `attesa_input` è `True`. Il mouse è attivo.
- **Animazione**: Appena clicchi, `attesa_input` diventa `False`. Il gioco elabora l'effetto e mostra i flash.
- **Turno Voldemort**: Dopo l'animazione, il computer sceglie una mossa e si torna alla fase di animazione.

🎨 Barre della Vita

Per i punti vita usiamo due variabili diverse: una per la logica, l'altra per la grafica che scorre piano per sembrare fluida: `punti_vita_harry` e `punti_vita_voldemort` (i PV reali)

`display_punti_vita_*` : valore usato per disegnare la barra (per animare)

In `update()`, se il valore visualizzato è maggiore di quello reale, lo facciamo scendere lentamente:

```
if display_punti_vita_harry > punti_vita_harry:
    display_punti_vita_harry -= 1  # Crea l'effetto "barra che scorre"
```

REMEMBER: Questo trucco rende il gioco molto più professionale e "smooth" (fluido) agli occhi del giocatore.

Combattimento: Scelta e Precisione

1. Prende i dati dell'incantesimo scelto (danno, precisione, nome)
2. Calcola se l'incantesimo va a segno usando `random.random()`

```
successo = random.random() < precisione
if successo:
    punti_vita_difensore -= danno
    messaggio = f"{attaccante} colpisce con {incantesimo}!"
else:
    messaggio = f"{incantesimo} è fallito!"
```

> ***Matematica del gioco**: Se un incantesimo ha precisione `0.8` (80%), abbiamo l'80% di probabilità che il numero generato sia minore di 0.8.*

🛡️ Logica di Combattimento: Danno

1. Se l'incantesimo va a segno:
    - Se `danno < 0` → cura
    - Altrimenti → sottrae PV al difensore e chiama `flash_danno`
2. Aggiorna `messaggio` e `descrizione`
3. Controlla se qualcuno è arrivato a 0 PV → pianifica `termina_gioco`

🖱️ UI Design: Input: mouse e tastiera

- `on_mouse_down(pos)` : controlla quale rettangolo è stato cliccato.
- `on_key_down(key)` : se il gioco è finito e premi SPAZIO, ricomincia (`reset_gioco`).

Nota pratica: i rettangoli si calcolano con matematica semplice (colonna/riga).

🖱️ UI Design: La Griglia 2x2

Dobbiamo disporre 4 tasti. Invece di scrivere 4 posizioni manuali, usiamo la matematica:

```
for i in range(len(opzioni_correnti)):
    # i può essere 0, 1, 2, 3
    x = 40 + (i % 2) * 380  # Resto della divisione: alterna tra 0 e 1 (colonne)
    y = 440 + (i // 2) * 60 # Divisione intera: 0,0 per i primi due, 1,1 per gli altri (righe)

    screen.draw.rect(Rect((x, y), (350, 50)), "white")
```

PRO TIP: Questa formula è universale. Cambiando il divisore (es. `% 3`) puoi creare griglie di qualsiasi dimensione (3x3, 4x4, ecc.) senza riscrivere il codice.

Animazioni: `animate()` e `clock`

Obiettivo: far sembrare che lo sprite subisca un colpo.

- Spostiamo lo sprite di qualche pixel con `animate()`
- Lo facciamo lampeggiare (cambia `opacity`): `clock.schedule_unique()` per far sparire e riapparire lo sprite velocemente quando subisce danni.
- Riportiamo la posizione originale

```
def flash_danno(sprite):
    # Cambiamo l'opacità: 0 = invisibile, 255 = visibile
    for i in range(3):
        clock.schedule_unique(lambda: setattr(sprite, "opacity", 0), i * 0.4)
        clock.schedule_unique(lambda: setattr(sprite, "opacity", 255), i * 0.4 + 0.2)
```

⚡ Cambio turno e `clock.schedule_unique`

- Dopo che Harry clicca un incantesimo, il gioco aspetta l'animazione.
- Se il gioco continua (nessun personaggio ha punti vita uguali a 0), dopo 3 secondi la funzione `fase_voldemort` viene eseguita.

Questo meccanismo evita che tutto succeda istantaneamente e dà tempo all'animazione.

📂 Struttura del Progetto

Assicurati che i tuoi file siano organizzati così per evitare errori `FileNotFound`:

- 📂 `progetto_duello/`
- 📄 `harry_potter.py` (Il tuo codice)
- 📄 `spells.csv` (Il database degli incantesimi)
- 📂 `images/` (Tutte le immagini `.png`)
- `harry.png`, `voldemort.png`, `hogwarts_bg.png` ...
- 📂 `sounds/` (Facoltativo: file `.wav` o `.ogg` per i colpi)

📌 Errori comuni e come risolverli

- `FileNotFoundError: spells.csv` → controlla il percorso e che il file sia nella cartella giusta.
- Errori di import → installa le librerie con `pip install pgzero polars` o prova `pandas`.
- `polars.exceptions.ColumnNotFoundError`: unable to find column …

💡 Idee di sviluppo

Prova a modificare il codice per aggiungere queste funzionalità:

1. **Critici**: Se un incantesimo colpisce, c'è una piccola probabilità (es. 10%) che faccia il doppio dei danni.
2. **Stamina**: Ogni incantesimo consuma "Energia Magica". Se finisce, devi saltare un turno per ricaricare.
3. **Logica Voldemort**: Rendi Voldemort più intelligente; invece di scegliere a caso, fagli usare una cura se ha poca vita!

📌 Conclusione

Il codice non è solo una lista di istruzioni, ma un insieme di **sistemi** (Dati, Grafica, Logica) che comunicano tra loro.

L'uso di strumenti come **Polars** ci permette di pensare come veri sviluppatori di giochi moderni, dove il contenuto (gli incantesimi) è separato dal motore di gioco.

> ***Bacchette pronte... al lavoro!***