RAGAZZE DIGITALI ER

Digital Girl Power

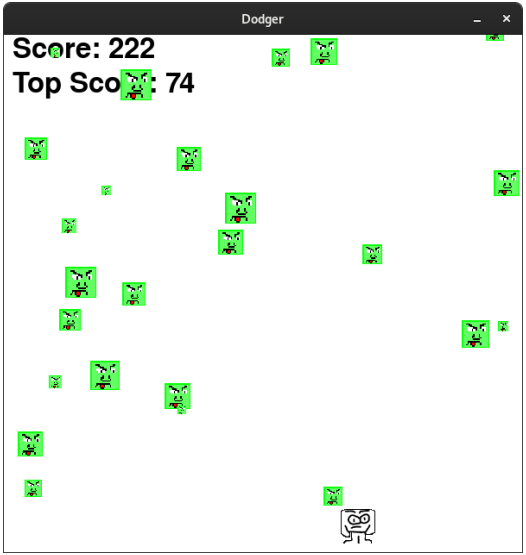
Cosa faremo oggi

Gioco finale

Oggi analizzeremo un gioco da cui potrete prendere esempio per i vostri progetti!
Step:

- scaricate il codice del gioco *download*;
- aprite il file con il codice su PyCharm;
- eseguite il gioco;
- commentate il codice (#commento in Python);
- riutilizzate il codice!

Dodger



Import dei moduli

```
1 import pygame, random, sys
2 from pygame.locals import *
```

Importiamo i moduli utili per creare un gioco (quelli che abbiamo già visto).

Variabili costanti

```
1 WINDOWWIDTH = 600
2 WINDOWHEIGHT = 600
3 TEXTCOLOR = (0, 0, 0)
4 BACKGROUNDCOLOR = (255, 255, 255)
5 FPS = 60
```

- dimensioni finestra;
- colore del testo e dello sfondo;
- numero di *frame* per secondo. Più è alto questo numero, più il gioco andrà veloce.

Variabili costanti

```
1 BADDIEMINSIZE = 10
2 BADDIEMAXSIZE = 40
3 BADDIEMINSPEED = 1
4 BADDIEMAXSPEED = 8
5 ADDNEWBADDIERATE = 6
6 PLAYERMOVERATE = 5
```

- dimensioni e velocità dei nemici;
- velocità con cui vengono creati nuovi nemici;
- velocità di movimento del giocatore.

Funzioni

```
1 def terminate():
2     pygame.quit()
3     sys.exit()
```

Per chiudere il gioco è necessario chiamare entrambe queste funzioni.
Può essere utile raggrupparle in un'unica funzione `terminate()`.

Funzioni

```
1 def waitForPlayerToPressKey():
2     while True:
3         for event in pygame.event.get():
4             if event.type == QUIT:
5                 terminate()
6             if event.type == KEYDOWN:
7                 if event.key == K_ESCAPE:
8                     terminate()
9                 return
```

Per mettere in pausa il gioco (inizio e Game Over) è possibile richiamare la funzione `waitForPlayerToPressKey()`. Questa funzione verifica se l'utente vuole chiudere la finestra (`QUIT` o `K_ESCAPE`) o se vuole continuare il gioco premendo qualunque altro tasto.

Funzioni

```
1 def playerHasHitBaddie(playerRect, baddies):
2     for b in baddies:
3         if playerRect.colliderect(b['rect']):
4             return True
5     return False
```

Per sapere se il giocatore si è scontrato con un cattivo si può utilizzare la funzione `playerHasHitBaddie(playerRect, baddies)`. Se il "rettangolo" del giocatore collide (si "tocca") con il rettangolo anche di un solo nemico allora la funzione restituisce True altrimenti False. I nemici nel gioco sono tanti quindi baddies è una lista.

Ricordate il metodo `colliderect(rect)`?

Funzioni

```
1 def drawText(text, font, surface, x, y):
2     textobj = font.render(text, 1, TEXTCOLOR)
3     textrect = textobj.get_rect()
4     textrect.topleft = (x, y)
5     surface.blit(textobj, textrect)
```

Per mostrare un testo (es. per il Game Over o per il punteggio) si può utilizzare la funzione `drawText(text, font, surface, x, y)`.
Come potete vedere, per prima cosa viene renderizzato il testo (ottenendo una superficie). Dopodiché vengono richieste le coordinate rettangolari e modificate affinché l'angolo in alto a sinistra coincida con le coordinate x,y passate. Infine viene "incollata" (ricordate il metodo `blit(...)`?) la superficie del testo (`textobj`) sulla superficie passata (`surface`), nelle coordinate rettangolari (`textrect`).

Inizializzazione di pygame e della finestra di gioco

```
1 # Set up pygame, the window, and the mouse cursor.
2 pygame.init()
3 mainClock = pygame.time.Clock()
4 windowSurface = pygame.display.set_mode((WINDOWWIDTH,
     WINDOWHEIGHT))
5 pygame.display.set_caption('Dodger')
6 pygame.mouse.set_visible(False)
```

Utilizzeremo la variabile `mainClock` per gestire il tempo del nostro gioco.
La variabile `windowSurface` rappresenta la finestra di gioco.
La funzione `pygame.display.set caption()` permette di modificare il titolo della finestra.
La funzione `pygame.mouse.set visible(False)` dice a `pygame` di "nascondere" il cursore.

Impostazione del font

```
1 # Set up the fonts.
2 font = pygame.font.SysFont(None, 48)
```

Passando `None` alla funzione `pygame.font.SysFont()` si utilizza il font di default. Al suo posto si può mettere qualsiasi altro font (Arial, Times New Roman,...)

Impostazione della musica e dei suoni

```
1 # Set up sounds.
2 gameOverSound = pygame.mixer.Sound('gameover.wav')
3 pygame.mixer.music.load('background.mid')
```

La variabile `gameOverSound` rappresenta la musichetta che il gioco riproduce quando il giocatore perde. La musica di sottofondo viene caricata con la funzione `pygame.mixer.music.load()`.
Al posto di gameover.wav e background.mid si possono scaricare e utilizzare altre musichette seguendo gli step seguenti:

- scaricare un file .wav o .midi da google;
- spostare il file scaricato nella directory del vostro gioco;
- cambiare il nome del file nel codice sorgente.

Impostazione delle immaginii

```
1 # Set up images.
2 playerImage = pygame.image.load('player.png')
3 playerRect = playerImage.get_rect()
4 baddieImage = pygame.image.load('baddie.png')
```

Come per i suoni potete utilizzare le vostre immagini per i personaggi del gioco. Queste immagini devono essere salvate nella directory del gioco e devono avere l'estensione .png o .jpeg. Infine ricordatevi di cambiare il nome dell'immagine nel codice.

Schermata iniziale

```
1 # Show the "Start" screen.
2 windowSurface.fill(BACKGROUNDCOLOR)
3 drawText('Dodger', font, windowSurface, (WINDOWWIDTH / 3),
     (WINDOWHEIGHT / 3))
4 drawText('Press a key to start.', font, windowSurface,
     (WINDOWWIDTH / 3) - 30, (WINDOWHEIGHT / 3) + 50)
5 pygame.display.update()
6 waitForPlayerToPressKey()
```

All'inizio del gioco vogliamo mostrare una schermata con il nome del gioco e la scritta "Premi un tasto per iniziare". Dopodiché ci mettiamo in attesa dell'azione richiamando la funzione `waitForPlayerToPressKey()`.

Inizio del gioco

Ora che abbiamo definito tutte le funzioni utili e *l'ambiente* di partenza possiamo iniziare a scrivere il nostro gioco!

Inizio del gioco

```
1  topScore = 0
2  while True:
3      # Set up the start of the game.
4      baddies = []
5      score = 0
6      playerRect.topleft = (WINDOWWIDTH / 2, WINDOWHEIGHT - 50)
7      moveLeft = moveRight = moveUp = moveDown = False
8      reverseCheat = slowCheat = False
9      baddieAddCounter = 0
10     pygame.mixer.music.play(-1, 0.0)
```

Inizio del gioco

- creiamo una variabile per salvare il punteggio migliore (inizialmente a zero);
- iniziamo il *loop* del gioco (gestisce partite multiple, una per ogni ciclo);
  - azzeriamo i nemici (lista di nemici vuota);
  - azzeriamo il punteggio della partita;
  - posizioniamo il giocatore al centro, in basso;
  - il giocatore è fermo (inizializziamo tutti i move*** a False);
  - i trucchi sono disabilitati (inizializziamo i ***Cheat a False);
  - azzeriamo il contatore usato per far nascere periodicamente i nemici;
  - facciamo partire la musica subito, in ripetizione.

Partita

```
1     while True:
2         score += 1 # Increase score.
```

Questo invece è il loop della partita (annidato nel loop di prima)! Qui dentro dobbiamo gestire l'avanzamento della singola partita proprio come abbiamo visto nelle scorse lezioni. In questo gioco il punteggio cresce con il tempo, incrementiamolo ad ogni ciclo.

Partita

Iniziamo a gestire gli eventi!

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

```
1             if event.type == QUIT:
2                 terminate()
```

Se premo...

...la “X” della finestra, termino il gioco chiamando la funzione `terminate()`.

Partita

```
1                if event.type == KEYDOWN:
2                    if event.key == K_z:
3                        reverseCheat = True
4                    if event.key == K_x:
5                        slowCheat = True
```

Se inizio a premere...

...il tasto **X** attivo il trucco “reverse” (i nemici si muoveranno al contrario);
...il tasto **Z** attivo il trucco “slow” (i nemici si muoveranno al lentamente).

Partita

```
                if event.key == K_LEFT or event.key == K_a:
                    moveRight = False
                    moveLeft = True
                if event.key == K_RIGHT or event.key == K_d:
                    moveLeft = False
                    moveRight = True
                if event.key == K_UP or event.key == K_w:
                    moveDown = False
                    moveUp = True
                if event.key == K_DOWN or event.key == K_s:
                    moveUp = False
                    moveDown = True
```

Se inizio a premere...

...le frecce (o i tasti WASD), gestisco il movimento come visto negli scorsi esempi.

Partita

```
1                if event.type == KEYUP:
2                    if event.key == K_z:
3                        reverseCheat = False
4                        score = 0
5                    if event.key == K_x:
6                        slowCheat = False
7                        score = 0
8                    if event.key == K_ESCAPE:
9                            terminate()
```

Se smetto di premere...

...i tasti **X** o **Z** disattivo il relativo trucco e azzero il punteggio;
...il tasto **ESC** termino il gioco chiamando la funzione `terminate()`.

Partita

Se smetto di premere...

...le frecce (o i tasti WASD), gestisco il movimento come visto negli scorsi esempi.

```
1                if event.type == MOUSEMOTION:
2                    playerRect.centerx = event.pos[0]
3                    playerRect.centery = event.pos[1]
```

Se muovo il mouse...

...sposto il giocatore esattamente in quella posizione.

Aggiungere i nemici

Ora che abbiamo terminato di gestire gli eventi andiamo avanti con la logica del gioco.

```
1         # Add new baddies at the top of the screen, if needed.
2         if not reverseCheat and not slowCheat:
3             baddieAddCounter += 1
```

Solo se **non** stiamo usando trucchi, incrementiamo il contatore `baddieAddCounter` ad ogni ciclo. Ricorda: un nuovo nemico deve nascere ogni qual volta questo contatore raggiunge il valore nella costante `ADDNEWBADDIERATE`.

Aggiungere i nemici

```
        if baddieAddCounter == ADDNEWBADDIERATE:
            baddieAddCounter = 0
            baddieSize = random.randint(BADDIEMINSIZE, BADDIEMAXSIZE)
            newBaddie = {'rect': pygame.Rect(random.randint(0, WINDOWWIDTH
                - baddieSize), 0 - baddieSize, baddieSize, baddieSize),
                        'speed': random.randint(BADDIEMINSPEED,
                            BADDIEMAXSPEED),
                        'surface': pygame.transform.scale(baddieImage,
                            (baddieSize, baddieSize)),
                        }

            baddies.append(newBaddie)
```

Se il contatore raggiunge la soglia, lo azzeriamo poi dobbiamo generare un nuovo nemico da aggiungere alla lista (posizione orizzontale, dimensione e velocità casuali). Le proprietà del nemico (il Rect, la velocità e l'immagine scalata) sono salvate in un dizionario.

Muovere il giocatore

```
1         # Move the player around.
2         if moveLeft and playerRect.left > 0:
3             playerRect.move_ip(-1 * PLAYERMOVERATE, 0)
4         if moveRight and playerRect.right < WINDOWWIDTH:
5             playerRect.move_ip(PLAYERMOVERATE, 0)
6         if moveUp and playerRect.top > 0:
7             playerRect.move_ip(0, -1 * PLAYERMOVERATE)
8         if moveDown and playerRect.bottom < WINDOWHEIGHT:
9             playerRect.move_ip(0, PLAYERMOVERATE)
```

Usiamo i booleani per verificare in che direzione spostare il giocatore come abbiamo visto negli scorsi esempi. Questa volta però usiamo il metodo `move_ip(dX, dY)` che sposta le coordinate (il Rect) delle quantità passate, rispettivamente su x e su y.

Muovere i nemici

```
1         # Move the baddies down.
2         for b in baddies:
3             if not reverseCheat and not slowCheat:
4                 b['rect'].move_ip(0, b['speed'])
5             elif reverseCheat:
6                 b['rect'].move_ip(0, -5)
7             elif slowCheat:
8                 b['rect'].move_ip(0, 1)
```

Ora muoviamo tutti i nemici.
Se non ci sono trucchi attivi muoviamoli verso il basso della loro velocità;
se, invece, **è attivo il trucco "reverse"** muoviamoli verso l'alto a velocità 5;
se, invece, **è attivo il trucco "slow"** muoviamoli verso il basso ma piano, a velocità 1.

Cancellare i nemici

```
1         # Delete baddies that have fallen past the bottom.
2         for b in baddies[:]:
3             if b['rect'].top > WINDOWHEIGHT:
4                 baddies.remove(b)
```

Quando i nemici “cadono” fuori dalla finestra di gioco (in basso) li eliminiamo rimuovendoli dalla lista.

Aggiornare la schermata

```
1         windowSurface.fill(BACKGROUNDCOLOR)
```

Come prima cosa coloriamo lo sfondo

```
1         # Draw the score and top score.
2         drawText('Score: ' + str(score), font, windowSurface, 10, 0)
3         drawText('Top Score: ' + str(topScore), font,
              windowSurface, 10, 40)
```

Punteggio

Disegniamo il punteggio corrente e il record grazie alla funzione `drawText(...)`.

Aggiornare la schermata

```
        # Draw the player's rectangle.
        windowSurface.blit(playerImage, playerRect)
        # Draw each baddie.
        for b in baddies:
            windowSurface.blit(b['surface'], b['rect'])
```

Giocatore e nemici

Disegniamo sia il giocatore che i nemici con la funzione `blit(...)` a cui passiamo le rispettive superfici (che contengono le immagini opportunamente scalate) e ile rispettive coordinate (i Rect).

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

Ricordiamoci di aggiornare il display per rendere visibili le nostre modifiche!

Gestire le collisioni e concludere il ciclo

```
        # Check if any of the baddies have hit the player.
        if playerHasHitBaddie(playerRect, baddies):
            if score > topScore:
                topScore = score # set new top score
            break
```

Se il giocatore ha colpito un nemico dobbiamo interrompere la partita (usciamo forzatamente dal ciclo `while` con il `break`) dopo aver eventualmente aggiornato il record.

```
        mainClock.tick(FPS)
```

Altrimenti proseguiamo con la partita, come ultima cosa dobbiamo chiamare il metodo `tick()`.

Game Over

```
1     # Stop the game and show the "Game Over" screen.
2     pygame.mixer.music.stop()
3     gameOverSound.play()
4
5     drawText('GAME OVER', font, windowSurface, (WINDOWWIDTH / 3),
          (WINDOWHEIGHT / 3))
6     drawText('Press a key to play again.', font, windowSurface,
          (WINDOWWIDTH / 3) - 80, (WINDOWHEIGHT / 3) + 50)
7     pygame.display.update()
8     waitForPlayerToPressKey()
9
10    gameOverSound.stop()
```

Game Over

Una volta terminata la partita (siamo usciti dal `while` più interno) dobbiamo mostrare la schermata di game over e poi dare la possibilità al giocatore di iniziare una nuova partita (ripartendo dall'inizio del `while` più esterno).

- fermiamo la musica e facciamo partire il suono di game over;
- disegnamo una scritta "Game Over" e una "Premi un tasto per giocare ancora";
- ricordiamoci di aggiornare il display;
- mettiamoci in attesa del giocatore chiamando `waitForPlayerToPressKey();`
- infine (il giocatore vuole giocare) fermiamo il suono di game over.

Ora il `while` più esterno ripartirà da capo inizializzando una nuova partita.

Consigli per il vostro progetto

Ora tocca a voi inventare e programmare un videogioco!
Alcuni consigli:

- riutilizzare le funzioni già viste (o qualsiasi altro codice che vi sembra utile);
- leggere bene gli eventuali errori;
- utilizzare la documentazione;
- cercare online;
- chiedere alle compagne e ai docenti.

Buon lavoro!

Materiale rilasciato con licenza

Creative Commons - Attributions, Share-alike 4.0