ragazze DIGITALI

IDEE PER UN FUTURO SMART

Outline

La finestra di gioco

```
WINDOW_WIDTH = 400
WINDOW_HEIGHT = 400
windowSurface =
    pygame.display.set_mode((WINDOW_WIDTH,
    WINDOW_HEIGHT), 0, 32)
pygame.display.set_caption('Input')
```

set_mode(..)

- La funzione set_mode(resolution, flags, depth) inizializza una nuova finestra
- I suoi parametri:
  - resolution
  - flags
  - depth

La finestra di gioco

set_mode(..)

- La funzione set_mode(resolution, flags, depth) inizializza una nuova finestra
- I suoi parametri:
  - resolution: una coppia di numeri nella forma (0, 0) che rappresenta rispettivamente la larghezza e l'altezza della finestra
  - flags: opzioni aggiuntive, alcuni esempi:
    pygame.FULLSCREEN create a fullscreen display
    pygame.RESIZABLE display window should be sizeable
  - depth: profondita' di colori. Possiamo non preoccuparcene, se settato a 0 pygame decide il valore migliore per noi.

pygame.Rect(...)

disegna un rettangolo e lo memorizza nella variabile player.
Quando dovremo cambiare il rettangolo che rappresenta il giocatore andremo a modificare questa variabile

pygame.Rect(...)

pygame.Rect(X_POSITION, Y_POSITION, WIDTH, HEIGHT)

- X_POSITION: Posizione della coordinata x del rettangolo sulla finestra
- Y_POSITION: Posizione della coordinata y del rettangolo sulla finestra
- WIDTH: Larghezza del rettangolo
- HEIGHT: Altezza del rettangolo

```
X_POSITION = 300
Y_POSITION = 100
PLAYER_WIDTH = 50
PLAYER_HEIGHT = 50
player = pygame.Rect(X_POSITION, Y_POSITION,
                     PLAYER_WIDTH, PLAYER_HEIGHT)
```

pygame.Rect(...)

Creiamo un altro rettangolo, come abbiamo fatto per player. Questa volta piu' piccolo.

Come possiamo fare per...

dare al nuovo rettangolo una posizione (X, Y) random?
Cosa devo scrivere al posto dei ??? (ricordiamoci di aggiungere import random all'inizio del nostro file)

```
foodSize = 20
# In questo caso non e' una costante perche'
   dopo voglio modificarne il valore
food = pygame.Rect(???, ???, foodSize, foodSize)
```

Qual e' la posizione (x, y) di food?

```
print(food.x, food.y)
```

Outline

Pygame puo' generare eventi in seguito a input da tastiera da parte dell'utente.

Eventi

- QUIT: Evento generato quando il giocatore chiude la finestra di gioco.
- KEYDOWN: Evento generato quando il giocatore tiene premuto un tasto qualsiasi.
- KEYUP: Evento generato quando il giocatore rilascia un tasto qualsiasi che precedentemente stava tenendo premouto.
- MOUSEMOTION: Evento generato quando il mouse si muove all'interno della nostra finestra.
- MOUSEBUTTONDOWN: Evento generato quando viene premuto un tasto del mouse all'interno della nostra finestra
- MOUSEBUTTONUP: Evento generato quando viene premuto un tasto del mouse all'interno della nostra finestra

Outline

Input da tastiera

Settiamo 4 variabili, che rappresentano le diverse posizioni, con valori booleani inizialmente False.
Quando il giocatore usera' il tasto sinistra cambieremo il valore corrispondente in True (e faremo lo stesso per gli altri tre tasti).
Quando il giocatore lascera' il tasto premuto setteremo di nuovo il valore corrispondente a False

```
moveLeft = False
moveRight = False
moveUp = False
moveDown = False
```

Gestire l'evento KEYDOWN (dentro il while True:)

Se l'evento scatenato e' KEYDOWN allora l'evento stesso ha un attributo chiamato "key" che indica quale tasto e' stato premuto.

```
for event in pygame.event.get():
    if event.type == KEYDOWN:
        if event.key == K_LEFT:
            moveRight = False
            moveLeft = True
        if event.key == K_RIGHT:
            moveLeft = False
            moveRight = True
        if event.key == K_UP:
            moveDown = False
            moveUp = True
        if event.key == K_DOWN:
            moveUp = False
            moveDown = True
```

Table 19-2: Constant Variables for Keyboard Keys

pygame constant variable	Keyboard key	pygame constant variable	Keyboard key
K_LEFT	Left arrow	K_HOME	HOME
K_RIGHT	Right arrow	K_END	END
K_UP	Up arrow	K_PAGEUP	PGUP
K_DOWN	Down arrow	K_PAGEDOWN	PGDN
K_ESCAPE	ESC	K_F1	F1
K_BACKSPACE	Backspace	K_F2	F2
K_TAB	TAB	K_F3	F3
K_RETURN	RETURN or ENTER	K_F4	F4
K_SPACE	Spacebar	K_F5	F5
K_DELETE	DEL	K_F6	F6
K_LSHIFT	Left SHIFT	K_F7	F7
K_RSHIFT	Right SHIFT	K_F8	F8
K_LCTRL	Left CTRL	K_F9	F9
K_RCTRL	Right CTRL	K_F10	F10
K_LALT	Left ALT	K_F11	F11
K_RALT	Right ALT	K_F12	F12

Gestire l'evento KEYUP

Se l'evento scatenato e' KEYUP il nostro quadrato si dovra' fermare.
Settiamo quindi la direzione del tasto rilasciato a False.

```
if event.type == KEYUP:
    if event.key == K_ESCAPE:
        pygame.quit()
        sys.exit()
    if event.key == K_LEFT:
        moveLeft = False
    if event.key == K_RIGHT:
        moveRight = False
    if event.key == K_UP:
        moveUp = False
    if event.key == K_DOWN:
        moveDown = False
```

Outline

Muovere il giocatore

Muovere il giocatore

Se una delle 4 variabili, moveDown, moveUp, moveLeft o moveRight ha valore True dobbiamo muovere il nostro giocatore (memorizzato nella variabile player).
In particolare possiamo muovere il giocatore verso l'alto se questo non e' gia' arrivato all'estremo superiore della finestra, in basso se non e' arrivato a quello inferiore e cosi' via..
Per rappresentare il movimento cambieremo la posizione verso la quale ci vogliamo muovere aggiungendo o sottraendo un valore $\mathrm{MOVE}_S PEED(cherappresentailquantocistiamomuovendo)$

```
if moveDown and player.bottom < WINDOW_HEIGHT:
    player.top = player.top + MOVE_SPEED
if moveUp and player.top > 0:
    player.top = player.top - MOVE_SPEED
```

Muovere il giocatore

Muovere il giocatore

Per rappresentare il movimento cambieremo la posizione verso la quale ci vogliamo muovere aggiungendo o sottraendo un valore $\mathrm{MOVE}_SPEED(cherappresentailquantocistiamomuovendo)$.
Dobbiamo inizializzare la costante $\mathrm{MOVE}_SPEED(all'iniziodelnostroprogramma, doveabbiamoinizializzat$

```
MOVE_SPEED = 6
```

Muovere il giocatore

Come possiamo fare per...

aggiungere il movimemnto verso destra e verso sinistra? Cosa devo scrivere al posto dei ???

```
if moveLeft and ???:
    player.left ??? ???
if moveRight and ???:
    player.right ??? ???
```

Outline

Disegnare il giocatore e il cibo

Per adesso abbiamo solo detto, definendo player, quali sono le caratteristiche del nostro giocatore.
Non lo abbiamo ancora disegnato e mostrato a video

```
windowSurface.fill(WHITE) # A cosa serve qui?
pygame.draw.rect(windowSurface, BLACK, player)
pygame.draw.rect(windowSurface, GREEN, food)
```

Outline

Collision detection

player.colliderec()

Andiamo a rilevare le collisioni tra il giocatore (player) e i cibi (foods). Utilizziamo la funzione colliderect che verifica se due rettangoli si toccano.

Collision Detection

```
if player.colliderect(food):
    # Faccio qualcosa.
    pygame.draw.rect(windowSurface, RED, food)
```

O se avessimo tanti "cibi" memorizzati in una lista (foods)

```
new_foods = foods
for food in new_foods:
    if player.colliderect(food):
        foods.remove(food)
new_foods = []
```

Outline

pygame.display.update()

pygame.display.update()

Le proprieta' del nostro giocatore sono cambiate, come anche quelle dei cibi. Il giocatore e' stato spostato, alcuni cibi rimossi e alcuni aggiunti. E' necessario aggiornare la finestra di gioco con le nuove caratteristiche di giocatore e cibi.

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

Clock

Clock

Inizializziamo, all'inizio del nostro programma, una variabile globale chiamata mainClock che ci aiutera' (in generale) a tenere traccia del passare del tempo.

```
mainClock = pygame.time.Clock()
```

pygame.time.tick(..)

pygame.time.tick(..)

come ultima riga del while utilizziamo la funzione tick(..) che, lascia scorrere del tempo tra una chiamata e l'altra della funzione stessa. Quindi, secondo l'esempio, dopo il primo passeranno 40 millisecondi prima che venga eseguito di nuovo il tick.

```
mainClock.tick(40)
```

Perche' questo?
Proviamo a togliere la riga mainClock.tick(40)

Clock

Clock

Creiamo ora un timer che tenga traccia del tempo trascorso e lo stampi sulla console

```
# All'inizio del nostro codice
startTime = pygame.time.get_ticks()

# Alla fine del nostro codice, come ultima riga
    del While True:
elapsedTime = pygame.time.get_ticks()
print('Tempo trascorso:',
      int(((elapsedTime - startTime)/1000)))
```

Outline

Input da mouse

Gestire eventi del mouse

- MOUSEMOTION: Evento generato quando il mouse si muove all'interno della nostra finestra
  Ha degli attributi. In particolare ci sara' utile **pos**: Indica la posizione del mouse all'interno della finestra. La posizione e' rappresentata come tupla (x, y) dove x e y sono le coordinate del mouse.
- MOUSEBUTTONDOWN: Evento generato quando viene premuto un tasto del mouse all'interno della nostra finestra
  Ha degli attributi. In particolare ci sara' utile **button**: Indica quale tasto del mouse e' stato premuto.
  1 rappresenta il tasto sinistro, 2 il tasto centrale (se presente), 3 il tasto destro, 4 se la rotella e' stata mossa verso l'alto o 5 se mossa verso il basso.
- MOUSEBUTTONUP: Ha gli stessi attributi di MOUSEBUTTONDOWN.

Esempio di eventi provenienti dal mouse

Dentro al for event in pygame.event.get() (che si trova dentro al while True:)

```
if event.type == MOUSEMOTION:
    print(event.pos[0], event.pos[1])
if event.type == MOUSEBUTTONUP:
    # Aumentiamo di 5 la dimensione di'food'
if event.type == MOUSEBUTTONDOWN:
    if event.button == 1:
        print('Hai premuto il tasto sinistro del
            mouse')
    elif event.button == 3:
        print('Hai premuto il tasto destro del
            mouse')
    else:
        print("Hai premuto il tasto",
            event.button)
```

Outline

Immagini

Andiamo ora a sostituire il nostro quadrato nero (per il giocatore) e verde (per il cibo) rispettivamente con l'icona di pacman e un pezzo di pizza. Imposteremo anche un'immagine di sfondo per il background del nostro gioco.

pygame.image.load(...)

- prende come parametro una stringa con il nome dell'immagine.
- ricordiamoci che l'immagine che vogliamo usare si deve trovare nella stessa cartella del nostro file python.

Immagini

pygame.image.load(...)

- Carica un'immagine a partire da un file specificato come parametro.
- Prende come parametro una stringa con il nome dell'immagine.
- Ricordiamoci che l'immagine che vogliamo usare si deve trovare nella stessa cartella del nostro file python.

```
backgroundImage =
    pygame.image.load('background.png')
playerImage = pygame.image.load('pacman.png')
foodImage = pygame.image.load('pizza.png')
```

Download background.png
Download pacman.png
Download pizza.png

Immagini

pygame.transform.scale(image, (width, height))

- Ridimensiona un'immagine
- Prende come parametri
  - l'immagine da ridimensionare
  - Una tupla (width, height) che rappresenta la larghezza e l'altezza desiderata per quell'immagine.

```
backgroundStretchedImage =
    pygame.transform.scale(backgroundImage,
    (WINDOW_WIDTH, WINDOW_HEIGHT))
playerStretchedImage =
    pygame.transform.scale(playerImage,
    (PLAYER_WIDTH, PLAYER_HEIGHT))
foodStretchedImage =
    pygame.transform.scale(foodImage, (foodSize,
    foodSize))
```

Immagini

Cosa faceva windowSurface.blit(...)???

Disegna un'immagine dentro una superficie (o una superficie dentro una superficie).
Non ci servono piu' ne windowSurface.fill(WHITE) ne i due pygame.draw.rect(...)

```
# windowSurface.fill(WHITE)
windowSurface.blit(backgroundStretchedImage,
    windowSurfaceRectangle)
windowSurface.blit(playerStretchedImage, player)
windowSurface.blit(foodStretchedImage, food)
#pygame.draw.rect(windowSurface, BLACK, player)
#pygame.draw.rect(windowSurface, GREEN, food)
```

Esempi

Con un solo food e le immagini - Download
Simile ma con una lista di cibi (foods) ma senza immagini - Download

Materiale rilasciato con licenza

Creative Commons - Attributions, Share-alike 4.0